

NSWCDD 84-371

REVISION 3

AD-A275 405



DTIC
ELECTE
FEB 4 1994
S C D

12

STATISTICAL MODELING AND ESTIMATION OF RELIABILITY FUNCTIONS FOR SOFTWARE (SMERFS) LIBRARY ACCESS GUIDE

BY DR. WILLIAM H. FARR

SYSTEMS RESEARCH AND TECHNOLOGY DEPARTMENT

OLIVER D. SMITH

EG&G, WASHINGTON ANALYTICAL SERVICES CENTER, INCORPORATED

DAHLGREN, VIRGINIA 22448

REVISED SEPTEMBER 1993

Approved for public release; distribution is unlimited.



NAVAL SURFACE WARFARE CENTER

DAHLGREN DIVISION

Dahlgren, Virginia 22448-5000

14296 94-03958



94 2 03 172

NSWCDD 84-371

REVISION 3

STATISTICAL MODELING AND ESTIMATION OF RELIABILITY FUNCTIONS FOR SOFTWARE (SMERFS) LIBRARY ACCESS GUIDE

BY DR. WILLIAM H. FARR

SYSTEMS RESEARCH AND TECHNOLOGY DEPARTMENT

OLIVER D. SMITH

EG&G, WASHINGTON ANALYTICAL SERVICES CENTER, INCORPORATED

DAHLGREN, VIRGINIA 22448

REVISED SEPTEMBER 1993

DTIC QUALITY INSPECTED 8

Approved for public release, distribution is unlimited.

NAVAL SURFACE WARFARE CENTER
DAHLGREN DIVISION
Dahlgren, Virginia 22448-5000

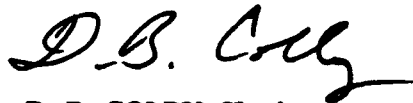
Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

FOREWORD

This report revises and supersedes NAVSWC TR 84-371, Revision 2, dated March 1991 (which should be discarded). The revision reflects the current Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) Library (SMFLIB). The library has undergone many enhancements and modifications, including (1) the implementation of the new model applicability analyses, such as accuracy, bias, trend, and noise; (2) the addition of the Jelinski/Moranda De-Eutrophication model for execution time data; (3) the update to the Schneidewind model for interval data to be more consistent with his current theories; and (4) the addition of a Goodness-of-Fit routine for the execution time models.

This report has been reviewed by Dr. Richard Lorey, Head, Advanced Computation Technology Group. Comments concerning this technical report should be directed to the Commander, NSWCDD, Attn: B10, Dahlgren, Virginia 22448-5000.

Approved by:

A handwritten signature in dark ink, appearing to read "D. B. Colby", is written over the printed name.

D. B. COLBY, Head
Systems Research and Technology Department

ABSTRACT

This is the second in a series of Naval Surface Warfare Center Dahlgren Division (NSWCDD) technical reports concerning software reliability. The first report, A Survey of Software Reliability Modeling and Estimation, NSWC TR 82-171, discusses various approaches advocated for reliability estimation; reviews various models proposed for this estimation process; provides model assumptions, estimates of reliability, and the precision of those estimates; and provides the data required for the models' implementation. Eight software reliability models were selected to form the basis of a library. This library also contains data edit, transformation, general statistics, and Goodness-of-Fit functions. The original Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) Library was described in the 1985 version of this report. The enhanced library, which now contains 11 models and model applicability analyses, is explained herein. The SMERFS User's Guide, NSWCDD TR 84-373, Revision 3, explains the execution of the new SMERFS driver.

CONTENTS

Chapter		Page
1	DOCUMENT OVERVIEW	1-1
	1.1 INTRODUCTION	1-1
	1.2 OPERATIONAL ENVIRONMENT	1-2
	1.3 DOCUMENT ORGANIZATION	1-3
	1.4 STANDARDIZED EXAMPLE DATA SETS	1-3
	1.5 ERROR DETECTION FACILITIES	1-6
2	GEOMETRIC MODEL FOR EXECUTION TIME DATA	2-1
	2.1 INTRODUCTION	2-1
	2.2 ACCESS LINES AND ARGUMENT LISTS	2-2
	2.3 EXAMPLES	2-5
3	JELINSKI/MORANDA DE-EUTROPHICATION MODEL	3-1
	3.1 INTRODUCTION	3-1
	3.2 ACCESS LINES AND ARGUMENT LISTS	3-2
	3.3 EXAMPLES	3-5
4	LITTLEWOOD AND VERRALL'S BAYESIAN RELIABILITY GROWTH MODEL FOR EXECUTION TIME DATA	4-1
	4.1 INTRODUCTION	4-1
	4.2 ACCESS LINES AND ARGUMENT LISTS	4-3
	4.3 EXAMPLES	4-6
5	JOHN MUSA'S BASIC EXECUTION TIME MODEL AND CALENDAR TIME COMPONENT	5-1
	5.1 INTRODUCTION	5-1
	5.2 ACCESS LINES AND ARGUMENT LISTS	5-3
	5.3 EXAMPLES	5-8
6	JOHN MUSA'S LOGARITHMIC POISSON EXECUTION TIME MODEL	6-1
	6.1 INTRODUCTION	6-1
	6.2 ACCESS LINES AND ARGUMENT LISTS	6-3
	6.3 EXAMPLES	6-6
7	NON-HOMOGENEOUS POISSON MODEL FOR EXECUTION TIME DATA ...	7-1
	7.1 INTRODUCTION	7-1
	7.2 ACCESS LINES AND ARGUMENT LISTS	7-3
	7.3 EXAMPLES	7-6

CONTENTS (Continued)

Chapter		Page
8	BROOKS AND MOTLEY'S DISCRETE SOFTWARE RELIABILITY MODEL FOR INTERVAL DATA	8-1
	8.1 INTRODUCTION	8-1
	8.2 ACCESS LINES AND ARGUMENT LISTS	8-2
	8.3 EXAMPLES	8-6
9	GENERALIZED POISSON MODEL FOR INTERVAL DATA	9-1
	9.1 INTRODUCTION	9-1
	9.2 ACCESS LINES AND ARGUMENT LISTS	9-2
	9.3 EXAMPLES	9-6
10	NON-HOMOGENEOUS POISSON MODEL FOR INTERVAL DATA	10-1
	10.1 INTRODUCTION	10-1
	10.2 ACCESS LINES AND ARGUMENT LISTS	10-2
	10.3 EXAMPLES	10-4
11	SCHNEIDEWIND'S MAXIMUM LIKELIHOOD MODEL FOR INTERVAL DATA	11-1
	11.1 INTRODUCTION	11-1
	11.2 ACCESS LINES AND ARGUMENT LISTS	11-2
	11.3 EXAMPLES	11-5
12	YAMADA'S S-SHAPED RELIABILITY GROWTH MODEL FOR INTERVAL DATA	12-1
	12.1 INTRODUCTION	12-1
	12.2 ACCESS LINES AND ARGUMENT LISTS	12-2
	12.3 EXAMPLES	12-4
13	UTILITY ROUTINE FOR DATA EDITS	13-1
	13.1 INTRODUCTION	13-1
	13.2 ACCESS LINE AND ARGUMENT LIST	13-1
	13.3 EXAMPLES	13-3
14	UTILITY ROUTINE FOR DATA TRANSFORMATIONS	14-1
	14.1 INTRODUCTION	14-1
	14.2 ACCESS LINE AND ARGUMENT LIST	14-1
	14.3 EXAMPLES	14-2

CONTENTS (Continued)

Chapter		Page
15	UTILITY ROUTINE FOR DATA STATISTICS	15-1
	15.1 INTRODUCTION	15-1
	15.2 ACCESS LINE AND ARGUMENT LIST	15-2
	15.3 EXAMPLES	15-2
16	UTILITY ROUTINE FOR MODEL FIT ANALYSIS	16-1
	16.1 INTRODUCTION	16-1
	16.2 ACCESS LINE AND ARGUMENT LIST	16-1
	16.3 EXAMPLES	16-2
17	REFERENCES	17-1

APPENDIXES

A	EXAMPLES FOR EXECUTION TIME DATA ANALYSES	A-1
B	EXAMPLES FOR INTERVAL DATA ANALYSES	B-1
C	EXAMPLES FOR UTILITY ROUTINE USAGES	C-1

DISTRIBUTION	(1)
--------------------	-----

TABLES

Table		Page
1-1	DATA FOR EXECUTION TIME ANALYSES	1-4
1-2	DATA FOR INTERVAL DATA ANALYSES	1-5
2-1	GEOMETRIC MODEL EXAMPLE PAGES	2-5
3-1	JELINSKI/MORANDA DE-EUTROPHICATION MODEL EXAMPLE PAGES ...	3-6
4-1	BAYESIAN RELIABILITY GROWTH MODEL EXAMPLE PAGES	4-7
5-1	MUSA'S BASIC EXECUTION TIME MODEL EXAMPLE PAGES	5-8
6-1	MUSA'S LOGARITHMIC POISSON MODEL EXAMPLE PAGES	6-6
7-1	NON-HOMOGENEOUS POISSON MODEL (EXECUTION TIME) EXAMPLE PAGES	7-6
8-1	DISCRETE SOFTWARE RELIABILITY MODEL EXAMPLE PAGES	8-7
9-1	GENERALIZED POISSON MODEL EXAMPLE PAGES	9-7
10-1	NON-HOMOGENEOUS POISSON MODEL (INTERVAL DATA) EXAMPLE PAGES	10-4

TABLES (Continued)

Table		Page
11-1	SCHNEIDEWIND'S MAXIMUM LIKELIHOOD MODEL EXAMPLE PAGES . . .	11-5
12-1	YAMADA'S S-SHAPED RELIABILITY GROWTH MODEL EXAMPLE PAGES .	12-4
13-1	DATA EDIT EXAMPLE PAGES	13-3
14-1	DATA TRANSFORMATION EXAMPLE PAGES	14-2
15-1	DATA STATISTICS EXAMPLE PAGES	15-2
16-1	MODEL FIT ANALYSIS EXAMPLE PAGES	16-2

CHAPTER 1

DOCUMENT OVERVIEW

1.1 INTRODUCTION

This report is designed to provide the required information for the use of the revised Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) Library (SMFLIB). The original SMFLIB contains the statistical and mathematical processing required to obtain various software reliability estimates found in Reference 1. The library was unique, in that it was the first library (to the author's knowledge) specifically designed for the analysis of software error data. Users who find it necessary to develop their own "tailored" software reliability analysis program [because the driver (Reference 2) does not fit into their organization] can do so by simply creating a driver. This driver will accept their error data, access the various estimation and prediction routines of the SMFLIB, and output the values.

In this manner, users can easily and quickly create a tailored software reliability program that can be useful in determining the readiness of the software for release, in determining allocation of testing personnel based upon estimated reliabilities of the components of the software, and aiding in determining when a program should be retired or rewritten. However, great care must be exercised in both the selection of models and the overall use of this methodology. Model assumptions and data requirements vary from model to model, as do the resulting predictions and their interpretations. There are many considerations to be addressed before attempting the development of a tailored driver. A good starting point for this stage is found in Reference 1.

In 1987, the first revision to the SMERFS software package was released. That revision primarily affected the driver and its documentation, but changes were also made to the SMFLIB. The most important modification was the discovery of an error in the implementation of the equations for the second and third treatment types of Schneidewind's Maximum Likelihood model for interval data analysis. That error was corrected in the release and cited in the first revision.

Another significant area of modification involved machine transportability. To transport the SMFLIB (as well as the new driver) from a mainframe computer [e.g., the Naval Surface Warfare Center Dahlgren Division (NSWCDD) Control Data Corporation (CDC) CYBER 170/875] to either a minicomputer (e.g., VAX 11/785) or any IBM-compatible Personal Computer (PC), only required changing the single precision declarations to double precision. The implicit variable declarations had to be modified to double precision when transporting to "smaller" computer systems to yield the same precision as on "larger" computer systems. The modification to the variable declaration line had to be performed in all routines of the driver and library.

In 1990, the second revision to the SMERFS software package was released. That revision primarily affected the library and its documentation. But, interface changes (caused by the new library)

and general enhancements were also made to the driver. Those changes included the consolidation of modules and the addition of an optional output file to interface with a (user-supplied) graphics program to produce high quality graphs of the data.

Prior to that release, the software reliability models included in the SMFLIB used either the Nelder-Mead algorithm or the Newton-Raphson procedure. Those numerical methods are more susceptible to the problem of "divergence," which is the functional value moving away from the maximum (minimum) value of the function (if one exists). The models could also encounter a possible division by zero, exponential overflow, or exponential underflow. Additionally, it was stressed in the previous SMFLIB documentation that the results of a particular model analysis should never be accepted until efforts were made to ensure that the global maximum (minimum) had been located.

The prior problems were almost completely eliminated by the use of the Dekker-Brent and Trust Region algorithms in the second revision. These optimization techniques establish most of the initial starting values, are more able to handle "misbehaving" data, and converge to the optimum value of the function more often and more readily (Reference 3). The models, which still required the user to perform several executions (to ensure that the global has been reached), contain comments to that effect in this NSWCDD technical report.

Two additional models were also added to the SMFLIB, bringing the total to 10. Those were John Musa's Logarithmic Poisson Execution model (Reference 4) and Yamada's S-Shaped Reliability Growth model (Reference 5, pages 475 through 478). John Musa's Basic Execution Time model was also updated to make it more consistent with his current software reliability models (Reference 4).

In 1993, the third revision to the SMERFS software package was released. This revision has equally affected the driver and the library. The changes to the driver include the reduction in size of the SMERFS data file to only the number of observed elements, the addition of the units for Time-Between-Failure (TBF) data types, the input of an American Standard Code for Information Interchange (ASCII) file of data (rather than requiring the user to input the initial data via the keyboard option), and the additional standardization of model estimations and predictions (through the addition of many equations within the driver using the parameters returned from the SMFLIB).

The changes to the SMFLIB (and the interface changes caused in the driver) include the addition of a Goodness-of-Fit analysis for execution time models through the Kolmogorov-Smirnov Statistic, the addition of the Jelinski/Moranda De-Eutrophication model, the addition of several model applicability analyses (Reference 6), and the alteration of the Schneidewind model to be more consistent with his current software reliability model (Reference 7).

1.2 OPERATIONAL ENVIRONMENT

The SMFLIB (as well as the SMERFS driver) is currently available on a number of different computers at NSWCDD, Dahlgren, Virginia. These range from large mainframes to PCs. The transfer to other computers should require no additional modifications or alterations, simply recompilation on the new target computer. The entire package is constructed using only a subset of the American National Standards Institute (ANSI) specifications for the FORTRAN 77 compiler. A FORTRAN 77 compiler (meeting those ANSI specifications) is the only requirement for establishing the program on another computer system, in addition to the previously discussed single and double precision declarations.

As for the establishment of the program on a PC, the authors are not aware of any IBM-compatible PC on which the program executable file cannot be simply loaded and executed. The SMFLIB (and the entire SMERFS software package) does not make use of any special hardware (e.g., a math coprocessor), special operating system features, or special peripheral support software (e.g., plotters or database managers).

1.3 DOCUMENT ORGANIZATION

This report is organized to provide easy access to the information about a particular model. Chapters 2 through 7 provide the information about the six software reliability models pertaining to execution time data. These models include:

- a. Geometric model
- b. Jelinski/Moranda De-Eutrophication model
- c. Littlewood and Verrall's Bayesian Reliability Growth model
- d. John Musa's Basic Execution Time model
- e. John Musa's Logarithmic Poisson model
- f. Non-homogeneous Poisson model (for execution time)

Chapters 8 through 12 provide the information for the interval data models. These models include:

- a. Brooks and Motley's Discrete Software Reliability model
- b. Generalized Poisson model
- c. Non-homogeneous Poisson model (for interval data)
- d. Schneidewind's Maximum Likelihood model
- e. Yamada's S-Shaped Reliability Growth model

As the SMFLIB attempts to provide the user with all routines critical for a complete software reliability analysis, the library also contains routines to edit a data vector, transform a data vector, determine the general statistics from a data vector, and perform a chi-square Goodness-of-Fit analysis for interval data. These additional routines are explained in Chapters 13 through 16.

This report also contains three appendixes that list the arguments before and after the SMFLIB routine processing. These lists were obtained from an actual SMERFS execution on the VAX 11/785, in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Appendix A contains the data lists for the execution time examples, Appendix B contains the data lists for the interval data examples, and Appendix C contains the data lists for the utility routine processing examples.

1.4 STANDARDIZED EXAMPLE DATA SETS

The example executions of the models in Appendixes A and B use "standardized" data sets. Here, the term standardized indicates that a single data set is used for execution time models, and a single data set is used for interval models. These data sets are listed in Tables 1-1 and 1-2.

The data in Table 1-1 are recorded in seconds of Central Processing Unit (CPU) execution time, and represent the times expended between failures. The first failure occurred 2 seconds into testing; the second failure occurred after an additional second (or a total of 3 seconds into testing); the third failure occurred after an additional 2 1/2 seconds (or a total of 5 1/2 seconds into testing). Although the data set reflects 30 data points, only 29 of these actually represent the execution time between adjacent failures. The final data point (number 30), which is enclosed in parentheses in the table, reflects the amount of failure-free execution time after the last failure (number 29).

TABLE 1-1. DATA FOR EXECUTION TIME ANALYSES

FAILURE NUMBER	TIME-BETWEEN-FAILURES	FAILURE NUMBER	TIME-BETWEEN-FAILURES
01	02.0	16	18.0
02	01.0	17	12.0
03	02.5	18	22.5
04	03.0	19	25.0
05	01.5	20	21.0
06	02.5	21	30.0
07	04.0	22	28.0
08	06.0	23	36.5
09	06.5	24	42.0
10	08.0	25	53.0
11	07.0	26	48.0
12	09.5	27	51.0
13	13.5	28	47.5
14	11.0	29	51.0
15	15.0	30	(30.0)

The data in Table 1-2 represent the number of faults occurring during 30 months of testing and the associated lengths (or testing intensities) of those testing intervals. The fourth column of Table 1-2 contains (randomly assigned) fractional testing lengths for use during the example execution of the Maximum Likelihood (ALPHA estimated) Generalized Poisson model (Chapter 9). These data are to reflect different testing intensity levels (i.e., a "2" means twice the effort was put into testing) in contrast to the "normal" scenario reflected by a "1".

TABLE 1-2. DATA FOR INTERVAL DATA ANALYSES

INTERVAL NUMBER	NUMBER OF FAULTS FOUND	TESTING LENGTHS	VARIABLE LENGTHS
01	20.0	1.0	0.25
02	18.0	1.0	0.50
03	22.0	1.0	0.75
04	22.0	1.0	1.00
05	17.0	1.0	0.75
06	21.0	1.0	0.50
07	17.0	1.0	0.25
08	19.0	1.0	0.50
09	17.0	1.0	0.75
10	17.0	1.0	1.00
11	15.0	1.0	0.75
12	18.0	1.0	0.50
13	14.0	1.0	0.25
14	12.0	1.0	0.50
15	12.0	1.0	0.75
16	10.0	1.0	1.75
17	08.0	1.0	0.75
18	05.0	1.0	1.00
19	07.0	1.0	1.00
20	03.0	1.0	1.00
21	00.0	1.0	1.00
22	03.0	1.0	0.75
23	02.0	1.0	1.00
24	03.0	1.0	0.75
25	01.0	1.0	1.00
26	00.0	1.0	1.00

TABLE 1-2. DATA FOR INTERVAL DATA ANALYSES (Continued)

INTERVAL NUMBER	NUMBER OF FAULTS FOUND	TESTING LENGTHS	VARIABLE LENGTHS
27	00.0	1.0	1.00
28	01.0	1.0	1.00
29	00.0	1.0	1.00
30	01.0	1.0	1.00

1.5 ERROR DETECTION FACILITIES

The SMFLIB routines perform internal error checks for potential error situations, which can generally be viewed as falling into two different categories. The first category includes routines in which the error condition terminates the routine prior to the performance of the requested processing. Here, the usual values of an error flag argument are either zero (signifying successful processing) or one (signifying an error condition occurred). For example, if the log of a non-positive number is requested using the SMFLIB routine for data transformations (Chapter 14), the error flag upon return has a value of one and the input data vector remains unchanged.

Within the second category, the returned error flag indicates the status of processing done within the subroutine. The return status flag is set to zero to indicate successful processing within the model being executed. In addition, the following five integer values are possible, where:

- a. One indicates that the maximum iteration count was reached before convergence occurred (during a Trust Region processing).
- b. Two indicates that the Trust Region could not be adjusted properly.
- c. Three indicates that the data are not appropriate for the model.
- d. Four indicates that the model estimates were deemed invalid (e.g., the model estimate for the total number of faults was less than the number of faults found to date).
- e. Five indicates that the model applicability analysis cannot be performed.

Under both types of error checking, it is the user's responsibility to examine the returned argument value and treat the returned data accordingly. The SMFLIB routines do not generate any error messages as such. Their only means of communication with the driver is through the call line.

CHAPTER 2

GEOMETRIC MODEL
FOR EXECUTION TIME DATA

2.1 INTRODUCTION

This chapter describes the SMFLIB routines associated with the Geometric model proposed by Paul Moranda (Reference 1). The three (main level) SMFLIB routines that support this model are SGEOMD, SGEOMA, and SGEOPR.

The first SMFLIB routine, SGEOMD, is used to calculate the estimates using both the Maximum Likelihood (ML) and Least Squares (LS) methods of execution. The ML estimates are obtained as the solution to the pair of equations given in Reference 1, page 4-53. The LS estimates are the solution to the pair of equations given in Reference 1, page 4-57. In both instances, the Dekker-Brent algorithm is used to solve the respective set of equations (Reference 3). The formulas for the variance of the ML estimates are given in Reference 1, pages 4-55 and 4-56. These are used to construct the approximate 95-percent confidence interval as:

$$\text{Estimate} \pm 1.95 * SD$$

where SD is equal to the standard deviation of the estimate.

The second SMFLIB routine, SGEOMA, is used to perform the four model applicability analyses (Reference 6) for the ML method of execution. This includes the analysis over accuracy, bias, trend, and noise. The accuracy of the model is calculated as:

$$ACC = \sum_{I=NSB}^{NSE} -\log(D) + I * \log(PCON) - D * PCON^I * DAT_{I,1}$$

where NSB and NSE are the user-specified starting and ending points for the iteration. (These values are usually set to one-half the number of failures and the number of failures minus one, respectively.) For each iteration, the SGEOMD routine is executed using I failures; and the estimates for the initial hazard rate (D) and the proportionality constant (PCON) based on those I failures are placed in the prior equation (along with the actual observed time to the next failure).

The vector for the bias of the model is calculated as:

$$U_I = 1.0 - \exp(-D * PCON^I * DAT_{I,1})$$

as I goes from NSB to NSE. Once all iterations are done, the vector is sorted in ascending order; and the Kolmogorov distance is calculated as:

$$DP = \max\left(0.0, \left(\frac{I}{NSR} - U_I\right), \left(U_I - \frac{I-1}{NSR}\right)\right)$$

as I goes from 1 to NSR (the range formed from NSB to NSE).

The calculation for the trend of the model starts with the same function as described for the bias; however, the values are scaled as:

$$T_I = -\text{LOG}(1.0 - U_I)$$

as I goes from NSB to NSE and then scaled a second time as:

$$Y_I = \sum_{k=1}^I \frac{T_k}{\text{SUMTI}}$$

where SUMTI is the summation of the T(I)'s.

The calculation for the noise of the model is calculated as:

$$\text{NSE} = \sum_{I=\text{NSB}}^{\text{NSE}} \left| \frac{N_I - N_{I-1}}{N_{I-1}} \right|$$

where M is calculated as:

$$N_I = \frac{-\text{LOG}(0.5)}{D * \text{PCON}^I}$$

The third SMFLIB routine, SGEOPR, is used to generate a vector of the predicted Time-Between-Failures (TBF) data. The Goodness-of-Fit of those predicted values to the original observed data is also computed. The estimate for the Ith failure (i.e., I-1 failures have been observed), where I ranges from one to NS, is calculated as:

$$\text{PDAT}_I = \frac{1.0}{D * \text{PCON}^{I-1}}$$

where D and PCON are estimates for the initial hazard rate and the proportionality constant of the model.

The Goodness-of-Fit calculation involves the same equations as shown for the bias calculation; however, the range goes from 1 to NS, and the values for D and PCON are calculated using the entire failure history.

2.2 ACCESS LINES AND ARGUMENT LISTS

2.2.1 SGEOMD Routine

The access line to the SGEOMD routine is as follows:

```
CALL SGEOMD (ESF      ,NS      ,DAT      ,STATS  ,RFLAG  )
```

where the arguments of the call line, in the order of occurrence, are defined as:

ESF = Input integer flag indicating the estimation selection type, where one indicates the ML method of execution and two indicates the LS method.

NS = Input integer containing the number of software failures observed to date.

- DAT** = Input real vector, of length greater than or equal to NS, containing the TBF data (in any consistent form of elapsed time measurement).
- STATS** = Output real array, of dimension four by three, containing the estimation results. These values should only be considered final when RFLAG has a value of zero. The first column of the STATS array contains the model estimates for:
- a. Proportionality constant
 - b. Initial hazard rate
 - c. Mean-Time-Before-Next-Failure (MTBNF)
 - d. Current purification level
- The second and third columns for the ML method contain the 95-percent confidence intervals for the associated estimates. The second and third columns are not used during the LS method and contain zeroes.
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing and three indicates the data are not appropriate for the model.

2.2.2 SGEOMA Routine

The access line to the SGEOMA routine is as follows:

```
CALL SGEOMA (DAT      ,NS      ,NSB      ,NSE      ,NSR      ,TYP
             ,STAT     ,RFLAG   ,INDX     ,V        ,VPRE     )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- DAT** = Input real vector, of length greater than or equal to NS, containing the TBF data (in any consistent form of elapsed time measurement).
- NS** = Input integer containing the number of software failures observed to date.
- NSB** = Input integer containing the starting point for the analysis.
- NSE** = Input integer containing the ending point for the analysis.
- NSR** = Input integer containing the range of the analysis.
- TYP** = Input integer flag indicating the type of analysis to be performed, where:
- a. One indicates the accuracy
 - b. Two indicates the bias
 - c. Three indicates the noise
 - d. Four indicates the trend

- STAT** = Output real containing the statistic for the analysis, where the following is returned (based on the assignment of the argument TYP):
- a. Computed accuracy
 - b. Computed Kolmogorov distance for the U-Plot
 - c. Computed noise
 - d. Computed Kolmogorov distance for the Y-Plot
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, and five indicates the model applicability analysis cannot be performed.
- INDX** = Output integer containing the number of iterations performed during the model applicability analysis.
- V** = Output real vector, of length greater than or equal to NSR, containing the data of the accuracy scatter-plot, bias U-Plot, or trend Y-Plot. The vector is not used for the noise analysis.
- VPRE** = Output real vector, of length greater than or equal to NSR, containing the data values for the bias U-Plot, prior to sorting. The vector is not used for the other three analysis types.

2.2.3 SGEOPR Routine

The access line to the SGEOPR routine is as follows:

```
CALL SGEOPR (D      ,PCON  ,NS      ,DAT    ,PDAT   ,DVAL
             ,DFLG   ,V      )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- D** = Input real containing the model estimate for the initial hazard rate [i.e., STATS(2,1) from the SGEOMD access].
- PCON** = Input real containing the model estimate for the proportionality constant [i.e., STATS(1,1) from the SGEOMD access].
- NS** = Input integer containing the number of software failures observed to date.
- DAT** = Input real vector, of length greater than or equal to NS, containing the TBF data (in any consistent form of elapsed time measurement). This argument is only used during the calculation of the Kolmogorov distance.
- PDAT** = Output real vector, of length greater than or equal to NS, containing the predicted data values.

- DVAL** = Output real containing the Kolmogorov distance.
- DFLG** = Output integer flag indicating the significance of the computed Kolmogorov distance, where one indicates the model may not provide an adequate fit and two indicates the model may provide an adequate fit.
- V** = Scratch working real vector of length greater than or equal to NS.

2.3 EXAMPLES

The example executions of the Geometric model are contained on pages A-3 through A-5 and consist of two analyses of the TBF data set introduced in Section 1.4; the first is for the ML and the second is for the LS. (That data set consists of 30 data points, with the last point being the amount of failure-free execution time after the last failure happened.) Table 2-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

The Geometric model does not have provisions for the final failure-free testing time; therefore, NS is set to 29 in both examples. [The failure-free testing time must, therefore, be considered for the prediction for the MTBNF in STATS(3,1).]

TABLE 2-1. GEOMETRIC MODEL EXAMPLE PAGES

METHOD OF EXECUTION (ESF)	ML	LS
MODEL APPLICABILITY ANALYSES	A-3	N/A
MODEL ESTIMATION EXECUTIONS	A-4	A-5
PREDICTED TBF VECTOR GENERATIONS	A-4	A-5

CHAPTER 3

JELINSKI/MORANDA
DE-EUTROPHICATION MODEL

3.1 INTRODUCTION

This chapter describes the SMFLIB routines associated with the Jelinski/Moranda De-Eutrophication model (Reference 1). The three (main level) SMFLIB routines that support this model are SJAMMD, SJAMMA, and SJAMPR.

The first SMFLIB routine, SJAMMD, is used to calculate the estimates using both the Maximum Likelihood (ML) and Least Squares (LS) methods of execution. The ML estimates are obtained as the solution to the pair of equations given in Reference 1, page 4-13. The LS estimates are the solution to the pair of equations given in Reference 1, page 4-14. In both instances, the Dekker-Brent algorithm is used to solve the respective set of equations (Reference 3).

The second SMFLIB routine, SJAMMA, is used to perform the four model applicability analyses (Reference 6) for the ML method of execution. This includes the analysis over accuracy, bias, trend, and noise. The accuracy of the model is calculated as:

$$ACC = \sum_{I=NSB}^{NSE} -\log(PCON) * \log(TNOF - I) - PCON * (TNOF - I) * DAT_{I+1}$$

where NSB and NSE are the user-specified starting and ending points for the iteration. (These values are usually set to one-half the number of failures and the number of failures minus one, respectively.) For each iteration, the SJAMMD routine is executed using I failures; and the estimates for the total number of faults (TNOF) and the proportionality constant (PCON) based on those I failures are placed in the prior equation (along with the actual observed time to the next failure).

The vector for the bias of the model is calculated as:

$$U_I = 1.0 - \exp(-PCON * (TNOF - I)) * DAT_{I+1}$$

as I goes from NSB to NSE. Once all iterations are done, the vector is sorted in ascending order; and the Kolmogorov distance is calculated as:

$$DP = \max\left(0.0, \left(\frac{I}{NSR} - U_I\right), \left(U_I - \frac{I-1}{NSR}\right)\right)$$

as I goes from 1 to NSR (the range formed from NSB to NSE).

The calculation for the trend of the model starts with the same function as described for the bias; however, the values are scaled as:

$$T_I = -\log(1.0 - U_I)$$

as I goes from NSB to NSE and then scaled a second time as:

$$Y_I = \sum_{K=1}^I \frac{T_K}{SUMTI}$$

where SUMTI is the summation of the T(I)'s.

The calculation for the noise of the model is calculated as:

$$NSE = \sum_{I=NSB}^{NSE} \left| \frac{M_I - M_{I-1}}{M_{I-1}} \right|$$

where M is calculated as:

$$M_I = \frac{-\text{LOG}(0.5)}{PCON * (TNOF - I)}$$

The third SMFLIB routine, SJAMPR, is used to calculate the Mean-Time-Before-Next-Failure (MTBNF) to discover the next K failures and to generate a vector of the predicted Time-Between-Failures (TBF) data. The Goodness-of-Fit of those predicted values to the original observed data is also computed during the generation of the predicted data vector. The estimate of the MTBNF to discover the next K failures is calculated as:

$$PDAT_I = \sum_{I=NS}^{NS+K-1} \frac{1.0}{PCON * (TNOF - I)}$$

where PCON is the estimate for the proportionality constant of the model, TNOF is the estimate for the total number of faults, NS is the current number of failures, and K is the additional number of failures to discover.

The estimate for the Ith failure (i.e., I-1 failures have been observed), where I ranges from one to NS, is calculated as:

$$PDAT_I = \frac{1.0}{PCON * (TNOF - I + 1)}$$

where PCON and TNOF are as previously defined.

The Goodness-of-Fit calculation involves the same equations as shown for the bias calculation; however, the range goes from 1 to NS, and the values for PCON and TNOF are calculated utilizing the entire failure history.

3.2 ACCESS LINES AND ARGUMENT LISTS

3.2.1 SJAMMD Routine

The access line to the SJAMMD routine is as follows:

```
CALL SJAMMD (ESF      ,NS      ,DAT      ,STATS  ,RFLAG  )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- ESF** = Input integer flag indicating the estimation selection type, where one indicates the ML method of execution and two indicates the LS method.
- NS** = Input integer containing the number of software failures observed to date.
- DAT** = Input real vector, of length greater than or equal to NS, containing the TBF data (in any consistent form of elapsed time measurement).
- STATS** = Output real vector, of length six, containing the estimation results. These values should only be considered final when RFLAG has a value of zero. The STATS vector contains the model estimates for:
- a. Proportionality constant
 - b. Initial hazard rate (or the initial intensity function)
 - c. Current hazard rate
 - d. Total number of faults
 - e. Total number of faults remaining
 - f. MTBNF
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, and four indicates the estimated number of faults is less than the number of faults found to date.

3.2.2 SJAMMA Routine

The access line to the SJAMMA routine is as follows:

```
CALL SJAMMA (DAT      ,NS      ,NSB      ,NSE      ,NSR      ,TYP
              ,STAT     ,RFLAG   ,INDX     ,V        ,VPRE     )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- DAT** = Input real vector, of length greater than or equal to NS, containing the TBF data (in any consistent form of elapsed time measurement).
- NS** = Input integer containing the number of software failures observed to date.
- NSB** = Input integer containing the starting point for the analysis.
- NSE** = Input integer containing the ending point for the analysis.
- NSR** = Input integer containing the range of the analysis.

- TYP** = Input integer flag indicating the type of analysis to be performed, where:
- a. One indicates the accuracy
 - b. Two indicates the bias
 - c. Three indicates the noise
 - d. Four indicates the trend
- STAT** = Output real containing the statistic for the analysis, where the following is returned (based on the assignment of the argument TYP):
- a. Computed accuracy
 - b. Computed Kolmogorov distance for the U-Plot
 - c. Computed noise
 - d. Computed Kolmogorov distance for the Y-Plot
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, four indicates the estimated number of faults is less than the number of faults found to date, and five indicates the model applicability analysis cannot be performed.
- INDX** = Output integer containing the number of iterations performed during the model applicability analysis.
- V** = Output real vector, of length greater than or equal to NSR, containing the data of the accuracy scatter-plot, bias U-Plot, or trend Y-Plot. The vector is not used for the noise analysis.
- VPRE** = Output real vector, of length greater than or equal to NSR, containing the data values for the bias U-Plot, prior to sorting. The vector is not used for the other three analysis types.

3.2.3 SJAMPR Routine

The access line to the SJAMPR routine is as follows:

```
CALL SJAMPR (NPV      ,NS      ,PCON      ,TNOF      ,EXE      ,DAT
             ,PDAT     ,DVAL     ,DFLG      ,V         )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- NPV** = Input integer containing the number of predicted values desired, where one indicates the MTBNF prediction and (the value of) NS indicates the vector of predicted TBF data.
- NS** = Input integer containing the number of software failures observed to date.

- PCON** = Input real containing the model estimate for the proportionality constant [i.e., STATS(1) from the SJAMMD access].
- TNOF** = Input real containing the model estimate for the total number of faults [i.e., STATS(4) from the SJAMMD access].
- EXE** = Input integer containing the desired number of failures to discover during the prediction for the MTBNF. This argument is not used during the vector generation.
- DAT** = Input real vector, of length greater than or equal to NS, containing the TBF data (in any consistent form of elapsed time measurement). This argument is only used during the calculation of the Kolmogorov distance.
- PDAT** = Output real vector, of length greater than or equal to NPV, containing the predicted data value(s).
- DVAL** = Output real containing the Kolmogorov distance.
- DFLG** = Output integer flag indicating the significance of the computed Kolmogorov distance, where one indicates the model may not provide an adequate fit and two indicates the model may provide an adequate fit.
- V** = Scratch working real vector of length greater than or equal to NS.

3.3 EXAMPLES

The example executions of the Jelinski/Moranda model are contained on pages A-6 through A-8 and consist of two analyses of the TBF data set introduced in Section 1.4; the first is for the ML and the second is for the LS. (That data set consists of 30 data points, with the last point being the amount of failure-free execution time after the last failure happened.) Table 3-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

It can be seen that the future prediction for the ML method was not executed. This is because the estimate for the total number of faults remaining in the program came out less than one for this data set.

The Jelinski/Moranda model does not have provisions for the final failure-free testing time; therefore, NS is set to 29 in both examples. [The failure-free testing time must, therefore, be considered for the prediction for the MTBNF in STATS(6) and MTBNF for the next K failures in PDAT(1).]

TABLE 3-1. JELINSKI/MORANDA DE-EUTROPHICATION MODEL EXAMPLE PAGES

METHOD OF EXECUTION (ESF)	ML	LS
MODEL APPLICABILITY ANALYSES	A-6	N/A
MODEL ESTIMATION EXECUTIONS	A-7	A-8
TOTAL MTBNF FOR THE NEXT K FAILURES	N/A	A-8
PREDICTED TBF VECTOR GENERATIONS	A-7	A-8

CHAPTER 4

**LITTLEWOOD AND VERRALL'S
BAYESIAN RELIABILITY GROWTH MODEL
FOR EXECUTION TIME DATA**

4.1 INTRODUCTION

This chapter describes the SMFLIB routines associated with the Bayesian Reliability Growth model proposed by Bev Littlewood and J. Verrall (Reference 1). The three (main level) SMFLIB routines that support this model are SLAVMD, SLAVMA, and SLAVPR.

The first SMFLIB routine, SLAVMD, is used to calculate the estimates using both the Maximum Likelihood (ML) and Least Squares (LS) methods of execution. The routine also provides for both linear and quadratic fits to the data, using the function:

$$B_0 + B_1 * I^{\Phi}$$

where BETA0 (B_0) and BETA1 (B_1) are parameters of the model and PHI (Φ) is set to one for the linear fit or to two for the quadratic fit.

The LS method gives an exact solution requiring no initial estimates or iterations. The ML method uses a Trust Region algorithm, requiring initial estimates (starting values) for the iteration process (References 1 and 3). The estimates from the LS method of execution provide reasonable initial estimates (for the Trust Region algorithm) for the ML estimates. However, there is no assurance that the global maximum will be reached using these values; therefore, several sets of starting values should be explored.

The second SMFLIB routine, SLAVMA, is used to perform the four model applicability analyses (Reference 6) for the ML method of execution. This includes the analysis over accuracy, bias, trend, and noise. The accuracy of the model is calculated as:

$$ACC = \sum_{I=NSB}^{NSE} -\log(A) + A * \log(TMP) - (A + 1.0) * \log(DAT_{I+1} + TMP)$$

where TMP is defined as:

$$TMP = B_0 + B_1 * (I+1)^{\Phi}$$

and NSB and NSE are the user-specified starting and ending points for the iteration. (These values are usually set to one-half the number of failures and the number of failures minus one, respectively.) For each iteration, the SLAVMD routine is executed using I failures; and the estimates for ALPHA (A), BETA0, and BETA1 based on those I failures are placed in the prior equations (along with the actual observed time to the next failure).

The vector for the bias of the model is calculated as:

$$U_I = 1.0 - \left(\frac{TMP}{DAT_{I+1} + TMP} \right)^A$$

as I goes from NSB to NSE (using ALPHA and TMP as defined for the accuracy). Once all iterations are done, the vector is sorted in ascending order; and the Kolmogorov distance is calculated as:

$$DP = \max \left(0.0, \left(\frac{I}{NSR} - U_I \right), \left(U_I - \frac{I-1}{NSR} \right) \right)$$

as I goes from 1 to NSR (the range formed from NSB to NSE).

The calculation for the trend of the model starts with the same function as described for the bias; however, the values are scaled as:

$$T_I = -\log(1.0 - U_I)$$

as I goes from NSB to NSE and then scaled a second time as:

$$Y_I = \sum_{k=1}^I \frac{T_k}{SUMTI}$$

where SUMTI is the summation of the T(I)'s.

The calculation for the noise of the model is calculated as:

$$NSE = \sum_{I=NSB}^{NSE} \left| \frac{M_I - M_{I-1}}{M_{I-1}} \right|$$

where M is calculated as:

$$M_I = \frac{TMP}{0.5^{\frac{1}{A}}} - TMP$$

(using ALPHA and TMP as defined for the accuracy).

The third SMFLIB routine, SLAVPR, is used to calculate the Mean-Time-Before-Next-Failure (MTBNF) to discover the next failure and to generate a vector of the predicted Time-Between-Failures (TBF) data. The Goodness-of-Fit of those predicted values to the original observed data is also computed during the generation of the predicted data vector. The estimate of the Ith failure (i.e., I-1 failures have been observed), where I ranges from one to NS, is calculated as:

$$PDAT_I = \frac{B_0 + B_1 + I^{\Phi}}{A - 1.0}$$

where ALPHA, BETA0, and BETA1 are estimates for the model parameters based upon ML or LS estimation; and PHI indicates whether the linear or quadratic fit was performed. Replacing I to the power PHI with NS+1 to the power PHI yields the equation to calculate the current MTBNF to discover the next failure.

The Goodness-of-Fit calculation involves the same equations as shown for the bias calculation; however, the range goes from 1 to NS, and the values for ALPHA, BETA0, and BETA1 are calculated utilizing the entire failure history.

4.2 ACCESS LINES AND ARGUMENT LISTS

4.2.1 SLAVMD Routine

The access line to the SLAVMD routine is as follows:

```
CALL SLAVMD (DAT      ,ESF      ,MAXIC  ,NS      ,N      ,PHIIND
              ,BETA    ,ALPHA    ,COUNT ,RFLAG   ,X      ,Z      )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- DAT** = Input real vector, of length greater than or equal to NS, containing the TBF data (in any consistent form of elapsed time measurement).
- ESF** = Input integer flag indicating the estimation selection type, where one indicates the ML method of execution and two indicates the LS method.
- MAXIC** = Input integer containing the maximum number of iterations to be performed before a return to the calling program is made if convergence does not occur. This argument is not used during the LS method of execution.
- NS** = Input integer containing the number of software failures observed to date.
- N** = Input integer containing the number of unknown parameters. It should currently be set to three.
- PHIIND** = Input integer flag indicating the PHI function to use, where one indicates the linear function and two indicates the quadratic function.
- BETA** = Input real vector, of length two, containing the initial estimates for BETA0 and BETA1 to be used during the ML method. This argument is not used during the LS method of execution. The estimates from the LS execution may be used as possible initial estimates (starting points); however, other starting points should also be used to ensure the global maximum is reached.

NOTE: Past experience with this model has shown that when BETA0 is less than BETA1, BETA0 should be changed to $2.0 * \text{PHIIND} * \text{BETA1}$ to aid in the search for the global maximum.

- ALPHA** = Output real containing the model estimate for ALPHA. The value should only be considered final when RFLAG has a value of zero. This argument returns with the constant two during the LS method.
- COUNT** = Output integer containing the number of iterations performed to obtain the model estimates during the ML method of execution. This argument is not used during the LS method of execution.

- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful convergence, one indicates the maximum iteration count was reached before convergence occurred, two indicates the Trust Region could not be adjusted properly, and four indicates that the model parameters failed to result in a positive initial MTBNF. (Improper adjustment may indicate that other starting points are required.)
- X** = Output real vector, of length two, containing the model estimates for BETA0 and BETA1. The values should only be considered final when RFLAG has a value of zero.
- Z** = Output real containing the optimum value of either the likelihood function or the sums-of-squares. The value should only be considered final when RFLAG has a value of zero.

4.2.2 SLAVMA Routine

The access line to the SLAVMA routine is as follows:

```
CALL SLAVMA (DAT      ,NS      ,NSB      ,NSE      ,NSR      ,TYP
             ,MAXIC    ,PHIIND ,STAT     ,RFLAG    ,INDX     ,V
             ,VPRE     )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- DAT** = Input real vector, of length greater than or equal to NS, containing the TBF data (in any consistent form of elapsed time measurement).
- NS** = Input integer containing the number of software failures observed to date.
- NSB** = Input integer containing the starting point for the analysis.
- NSE** = Input integer containing the ending point for the analysis.
- NSR** = Input integer containing the range of the analysis.
- TYP** = Input integer flag indicating the type of analysis to be performed, where:
- a. One indicates the accuracy
 - b. Two indicates the bias
 - c. Three indicates the noise
 - d. Four indicates the trend
- MAXIC** = Input integer containing the maximum number of iterations to be performed before a return to the calling program is made if convergence does not occur.

- PHIIND** = Input integer flag indicating the PHI function to use, where one indicates the linear function and two indicates the quadratic function.
- STAT** = Output real containing the statistic for the analysis, where the following is returned (based on the assignment of the argument TYP):
- a. Computed accuracy
 - b. Computed Kolmogorov distance for the U-Plot
 - c. Computed noise
 - d. Computed Kolmogorov distance for the Y-Plot
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, one indicates the maximum iteration count was reached before convergence occurred, two indicates the Trust Region could not be adjusted properly, four indicates that the model parameters failed to result in a positive initial MTBNF, and five indicates the model applicability analysis cannot be performed.
- INDX** = Output integer containing the number of iterations performed during the model applicability analysis.
- V** = Output real vector, of length greater than or equal to NSR, containing the data of the accuracy scatter-plot, bias U-Plot, or trend Y-Plot. The vector is not used for the noise analysis.
- VPRE** = Output real vector, of length greater than or equal to NSR, containing the data values for the bias U-Plot, prior to sorting. The vector is not used for the other three analysis types.

4.2.3 SLAVPR Routine

The access line to the SLAVPR routine is as follows:

```
CALL SLAVPR (NPV      ,NS      ,ALPHA  ,BETA0  ,BETA1  ,PHI
             ,DAT      ,PDAT    ,DVAL   ,DFLG   ,V      )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- NPV** = Input integer containing the number of predicted values desired, where one indicates the MTBNF prediction and (the value of) NS indicates the vector of predicted TBF data.
- NS** = Input integer containing the number of software failures observed to date.
- ALPHA** = Input real containing the model estimate for ALPHA (i.e., ALPHA from the SLAVMD access).

- BETA0** = Input real containing the model estimate for BETA0 [i.e., X(1) from the SLAVMD access].
- BETA1** = Input real containing the model estimate for BETA1 [i.e., X(2) from the SLAVMD access].
- PHI** = Input integer flag indicating whether the linear or quadratic function was used to obtain the estimates (i.e., PHIIND from the SLAVMD access).
- DAT** = Input real vector, of length greater than or equal to NS, containing the TBF data (in any consistent form of elapsed time measurement). This argument is only used during the calculation of the Kolmogorov distance.
- PDAT** = Output real vector, of length greater than or equal to NPV, containing the predicted data value(s).
- DVAL** = Output real containing the Kolmogorov distance.
- DFLG** = Output integer flag indicating the significance of the computed Kolmogorov distance, where one indicates the model may not provide an adequate fit and two indicates the model may provide an adequate fit.
- V** = Scratch working real vector of length greater than or equal to NS.

4.3 EXAMPLES

The example executions of the Bayesian Reliability Growth model are contained on pages A-9 through A-14 and consist of four analyses of the TBF data set introduced in Section 1.4; the first two are for the ML linear and quadratic executions and the second two are for the LS executions. (That data set consists of 30 data points, with the last point being the amount of failure-free execution time after the last failure happened.) Table 4-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

The Bayesian Reliability Growth model does not have provisions for the final failure-free testing time; therefore, NS is set to 29 in all examples. (The failure-free testing time must, therefore, be considered for the prediction for the MTBNF for the next failure.)

It can be seen that the future predictions and the vector generations were not performed for the both of the linear functions. This is because the model estimation executions were not successful, as indicated by the non-zero RFLAG values for this data set.

It can also be seen that the initial estimates for BETA0 and BETA1 in the ML examples were simply set to the estimates from the LS examples. Had this been a real software reliability analysis, other starting points should have been explored to ensure that the global maximum had been reached.

TABLE 4-1. BAYESIAN RELIABILITY GROWTH MODEL EXAMPLE PAGES

METHOD OF EXECUTION (ESF)	ML		LS	
SELECTION OF FUNCTION (PHIIND)	LIN	QUAD	LIN	QUAD
MODEL APPLICABILITY ANALYSES	A-9	A-11	N/A	N/A
MODEL ESTIMATION EXECUTIONS	A-10	A-12	A-13	A-14
MTBNF FOR THE NEXT FAILURE	N/A	A-12	N/A	A-14
PREDICTED TBF VECTOR GENERATIONS	N/A	A-12	N/A	A-14

CHAPTER 5

JOHN MUSA'S BASIC EXECUTION TIME MODEL AND CALENDAR TIME COMPONENT

5.1 INTRODUCTION

This chapter describes the SMFLIB routines associated with the Basic Execution Time model and Calendar Time Component developed by John Musa of Bell Laboratories (References 1 and 4). This model is based on the amount of Central Processing Unit (CPU) time involved in testing, but attempts to relate those estimates to wall clock units. By doing this, Musa is able to relate the amount of limiting resources (failure identification personnel, failure correction personnel, and computer time) to the amount of CPU time that may be utilized during various testing segments. Because of the extra relationship of CPU time to wall clock time, four (main level) SMFLIB routines are used to support this model. The four routines are SMUSMD, SMUSCT, SMUSMA, and SMUSPR.

The first SMFLIB routine, SMUSMD, is used to calculate the Maximum Likelihood (ML) estimates for the initial Mean-Time-Before-Next-Failure (MTBNF) and the required number of faults to be discovered before all faults in the code are uncovered. The two equations (Reference 1, page 4-87) are solved by the Dekker-Brent algorithm (Reference 3). The Basic Execution Time model (for this revision to the SMFLIB) has also been enhanced to include the additional quantities for the initial and current intensity functions (Reference 4), bringing the library more in line with Musa's current models.

The second SMFLIB routine, SMUSCT, provides the Calendar Time Component portion of Musa's model. In this routine, the CPU expenditure estimates are correlated to wall clock expenditure. Specifically, SMUSCT is used to calculate the prediction of the additional wall clock time required to achieve a desired CPU MTBNF (References 1 and 4).

The third SMFLIB routine, SMUSMA, is used to perform the four model applicability analyses (Reference 6) for the ML method of execution. This includes the analysis over accuracy, bias, trend, and noise. The accuracy of the model is calculated as:

$$ACC = \sum_{I=NSB}^{NSE} -\log(B_0) + \log(B_1) - TMP1 + B_0 * (EXP(-TMP1) - EXP(-TMP2))$$

where TMP1 and TMP2 are defined as:

$$TMP1 = B_1 * \sum_{J=1}^{I+1} DAT_J \qquad TMP2 = B_1 * \sum_{J=1}^I DAT_J$$

and NSB and NSE are the user-specified starting and ending points for the iteration. (These values are usually set to one-half the number of failures and the number of failures minus one, respectively.) For

each iteration, the SMUSMD routine is executed using I failures; and the estimates for BETA0 (B_0) and BETA1 (B_1) based on those I failures are placed in the prior equation (along with the actual observed time to the next failure). BETA0 is equivalent to the total number of faults (TNOF of the SMUSMD routine), BETA1 is re-computed as the initial intensity function (IIF of the SMUSMD routine) over the total number of faults, and the summations of the DAT(J)'s are the total amount of testing time that has been done up to the time of the current and next failure occurrences. DAT(J) is the time between the (J-1)st and Jth failure occurrences.

The vector for the bias of the model is calculated as:

$$U_I = 1.0 - \exp \left(- \left(B_0 * \exp \left(-B_1 * \sum_{j=1}^I \text{DAT}_j \right) \right) * (1.0 - \exp(-B_1 * \text{DAT}_{I+1})) \right)$$

as I goes from NSB to NSE. Once all iterations are done, the vector is sorted in ascending order; and the Kolmogorov distance is calculated as:

$$DP = \max \left(0.0, \left(\frac{I}{NSR} - U_I \right), \left(U_I - \frac{I-1}{NSR} \right) \right)$$

as I goes from 1 to NSR (the range formed from NSB to NSE).

The calculation for the trend of the model starts with the same function as described for the bias; however, the values are scaled as:

$$T_I = -\log(1.0 - U_I)$$

as I goes from NSB to NSE and then scaled a second time as:

$$Y_I = \sum_{k=1}^I \frac{T_k}{\text{SUMTI}}$$

where SUMTI is the summation of the T(I)'s.

The calculation for the noise of the model is calculated as:

$$NSE = \sum_{I=NSB}^{NSE} \left| \frac{M_I - M_{I-1}}{M_{I-1}} \right|$$

where M is calculated as:

$$M_I = -\frac{1.0}{B_1} * \log \left(1.0 + \frac{\log(0.5)}{B_0 * \exp \left(-B_1 * \sum_{j=1}^I \text{DAT}_j \right)} \right)$$

The fourth SMFLIB routine, SMUSPR, is used to calculate the predictions for the future reliability and additional testing time required to achieve a desired MTBNF and to generate a vector of predicted Time-Between-Failures (TBF) data. The Goodness-of-Fit of those predicted values to the original observed data is also computed during the generation of the predicted data vector. The estimate of the additional execution time required to increase the current MTBNF to a desired level is calculated as:

$$PDAT_2 = \text{TNOF} * \text{IMTBF} * \log \left(\frac{\text{DMTBF}}{\text{CMTBF}} \right)$$

where TNOF, IMTBF, and CMTBF are estimates for the total number of faults, initial MTBNF, and the current MTBNF, respectively. DMTBF is the desired level (References 1 and 4). Since a division by CMTBF is performed, this calculation should only be attempted if CMTBF is greater than zero.

The estimate of the predicted number of failures that need to occur before this reliability level is achieved, is calculated as:

$$PDAT_1 = \min \left((TNOF - NS), \left(TNOF * IMTBF * \left(\frac{1.0}{CMTBF} - \frac{1.0}{DMTBF} \right) \right) \right)$$

where TNOF, IMTBF, CMTBF, and DMTBF are as previously defined; and NS is the number of software failures observed to date.

The estimate for the I^{th} failure (i.e., $I-1$ failures have been observed), where I ranges from one to NS, is calculated as:

$$PDAT_I = IMTBF * \exp \left(\frac{\sum_{J=1}^{I-1} DAT_J}{TNOF * IMTBF} \right)$$

where TNOF and IMTBF are as previously defined, and the summation of the $DAT(J)$'s, where J ranges from one to $I-1$, is the total amount of testing time that has been done up to the time of the $(I-1)^{th}$ failure occurrence. $DAT(J)$ is the time between the $(J-1)^{th}$ and J^{th} failure occurrences.

The Goodness-of-Fit calculation involves the same equations as shown for the bias calculation; however, the range goes from 1 to NS, and the values for BETA0 and BETA1 are calculated utilizing the entire failure history.

5.2 ACCESS LINES AND ARGUMENT LISTS

5.2.1 SMUSMD Routine

The access line to the SMUSMD routine is as follows:

```
CALL SMUSMD (NS      ,DAT      ,XLTM      ,CMTBF      ,ECR      ,IMTBF
              ,IFLG     ,TNOF     ,IIF      ,CIF      ,RFLAG    )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- NS = Input integer containing the number of software failures observed to date.
- DAT = Input real vector, of length greater than or equal to NS, containing the CPU TBF data (in any consistent form of elapsed time measurement).
- XLTM = Input real containing the final failure-free test time. An entry of zero indicates that a software failure occurred at the end of the final entry (i.e., there is no failure-free test time).

- CMTBF** = Output real containing the model estimate of the current MTBNF. The value should only be considered final when RFLAG has a value of zero.
- ECR** = Output real containing the model estimate of the current reliability of the program if the program was to be run for the same amount of time as had been completed in testing. The value should only be considered final when RFLAG has a value of zero.
- IMTBF** = Output real vector, of length three, containing the model estimate of the initial MTBNF and (if IFLG is equal to one) its 95-percent confidence interval. The value(s) should only be considered final when RFLAG has a value of zero.
- IFLG** = Output integer flag indicating the availability of the 95-percent confidence interval within IMTBF, where zero indicates the interval was not computed and one indicates the interval was computed and stored in the second and third elements of IMTBF.
- TNOF** = Output real vector, of length three, containing the model estimate of the total number of faults and its 95-percent confidence interval. The values should only be considered final when RFLAG has a value of zero.
- IIF** = Output real containing the model estimate of the initial intensity function. The value should only be considered final when RFLAG has a value of zero.
- CIF** = Output real containing the model estimate of the current intensity function. The value should only be considered final when RFLAG has a value of zero.
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, and four indicates the estimated number of faults is less than the number of faults found to date.

5.2.2 SMUSCT Routine

The access line to the SMUSCT routine is as follows:

```
CALL SMUSCT (TNOF      ,CMTBFH ,DMTBFH ,IMTBFH ,NS      ,PC
             ,PF       ,PI      ,PMQ    ,RHOC   ,TC      ,TI
             ,XMC      ,XMF     ,XMI     ,XMQ    ,XINT    )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- TNOF** = Input real containing the model estimate for the total number of faults [i.e., TNOF(1) from the SMUSMD access].

- CMTBFH** = Input real containing the model estimate for the current MTBNF (recorded in hours) (i.e., CMTBF from the SMUSMD access, with possible time unit conversion).
- DMTBFH** = Input real containing the desired CPU MTBNF (recorded in hours).
- IMTBFH** = Input real containing the model estimate for the initial MTBNF (recorded in hours) [i.e., IMTBF(1) from the SMUSMD access, with possible time unit conversion].
- NS** = Input integer containing the number of software failures observed to date.
- PC** = Input real containing the number of computer shifts. For example, if the work week is 40 hours and the computer is available 80 hours per week, then there are 2 shifts.
- PF** = Input real containing the number of failure correction personnel.
- PI** = Input real containing the number of failure identification personnel.
- PMQ** = Input real containing the probability that the queue length will be no larger than the value assigned to the XMQ argument.
- RHOC** = Input real containing the computer utilization factor. RHOC should always be greater than zero and less than or equal to one.
- TC** = Input real containing the average amount of computer wall clock time expended per unit of CPU execution time.
- TI** = Input real containing the average amount of identification time expended per unit of CPU execution time.
- XMC** = Input real containing the average amount of computer time (in hours) expended per failure.
- XMF** = Input real containing the average amount of correction time (in hours) expended per failure.
- XMI** = Input real containing the average amount of identification time (in hours) expended per failure.
- XMQ** = Input real containing the maximum queue length for a debugger.
- XINT** = Output real containing the predicted wall clock time (in hours) required to achieve the goal specified in the DMTBFH argument.

5.2.3 SMUSMA Routine

The access line to the SMUSMA routine is as follows:

```
CALL SMUSMA (DAT      ,NS      ,NSB      ,NSE      ,NSR      ,TYP
             ,STAT     ,RFLAG   ,INDX     ,V        ,VPRE     )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- DAT** = Input real vector, of length greater than or equal to NS, containing the CPU TBF data (in any consistent form of elapsed time measurement).
- NS** = Input integer containing the number of software failures observed to date.
- NSB** = Input integer containing the starting point for the analysis.
- NSE** = Input integer containing the ending point for the analysis.
- NSR** = Input integer containing the range of the analysis.
- TYP** = Input integer flag indicating the type of analysis to be performed, where:
- a. One indicates the accuracy
 - b. Two indicates the bias
 - c. Three indicates the noise
 - d. Four indicates the trend
- STAT** = Output real containing the statistic for the analysis, where the following is returned (based on the assignment of the argument TYP):
- a. Computed accuracy
 - b. Computed Kolmogorov distance for the U-Plot
 - c. Computed noise
 - d. Computed Kolmogorov distance for the Y-Plot
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, four indicates the estimated number of faults is less than the number of faults found to date, and five indicates the model applicability analysis cannot be performed.
- INDX** = Output integer containing the number of iterations performed during the model applicability analysis.
- V** = Output real vector, of length greater than or equal to NSR, containing the data of the accuracy scatter-plot, bias U-Plot, or trend Y-Plot. The vector is not used for the noise analysis.

VPRE = Output real vector, of length greater than or equal to NSR, containing the data values for the bias U-Plot, prior to sorting. The vector is not used for the other three analysis types.

5.2.4 SMUSPR Routine

The access line to the SMUSPR routine is as follows:

```
CALL SMUSPR (NPV ,NS ,DAT ,IMTBF ,CMTBF ,DMTBF
             ,TNOF ,IIF ,PDAT ,DVAL ,DFLG ,V )
```

where the arguments of the call line, in the order of occurrence, are defined as:

NPV = Input integer containing the number of predicted values desired, where two indicates the predictions for future reliability and additional testing time, and (the value of) NS indicates the vector of predicted TBF data.

NS = Input integer containing the number of software failures observed to date.

DAT = Input real vector, of length greater than or equal to NS, containing the CPU TBF data (in any consistent form of elapsed time measurement). This argument is not used during the predictions for the future reliability and additional testing time.

IMTBF = Input real containing the model estimate for the initial MTBNF [i.e., IMTBF(1) from the SMUSMD access].

CMTBF = Input real containing the model estimate for the current MTBNF (i.e., CMTBF from the SMUSMD access). This argument is not used during the vector generation.

DMTBF = Input real containing the desired MTBNF. This argument is not used during the vector generation.

TNOF = Input real containing the model estimate for the total number of faults [i.e., TNOF(1) from the SMUSMD access].

IIF = Input real containing the model estimate for the initial intensity function (i.e., IIF from the SMUSMD access).

PDAT = Output real vector, of length greater than or equal to NPV, containing the predicted data values.

DVAL = Output real containing the Kolmogorov distance.

- DFLG** = Output integer flag indicating the significance of the computed Kolmogorov distance, where one indicates the model may not provide an adequate fit and two indicates the model may provide an adequate fit.
- V** = Scratch working real vector of length greater than or equal to NS.

5.3 EXAMPLES

The example executions of John Musa's Basic Execution Time model (and Calendar Time Component) are contained on pages A-15 through A-17 and consist of an analysis of the TBF data set introduced in Section 1.4. (That data set consists of 30 data points, with the last point being the amount of failure-free execution time after the last failure happened.) Table 5-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

It can be seen that the number of failures is set to 29 (in the example), and the XLTM argument is set to the final failure-free testing time of 30 seconds. Had the final testing session terminated with a failure, XLTM would have been assigned a zero, and NS would have been set to the number of data points (30). Within the model, the failure-free test time is removed after the model parameters are computed. This brings the estimates related to execution time more in line with the other models (which do not have provisions for the failure-free test time).

It is pointed out that the Calendar Time Component expects the initial and current MTBNF estimates to be recorded in hours. Since the data described in Table 1-1 are in seconds, the two values are divided by 3600. The desired MTBNF is set to 40 hours; the other inputs were simply set to reasonable values for the limiting resources (failure identification personnel, failure correction personnel, computer time). The result, XINT, indicates the amount of wall clock time (in hours) required to achieve the desired amount of CPU execution time (in hours) without failure.

TABLE 5-1. MUSA'S BASIC EXECUTION TIME MODEL EXAMPLE PAGES

MODEL APPLICABILITY ANALYSES	A-15
MODEL ESTIMATION EXECUTION	A-16
FUTURE RELIABILITY MEASURES AND ADDITIONAL TESTING TIME PREDICTIONS	A-16
PREDICTED TBF VECTOR GENERATION	A-16
CALENDAR TIME COMPONENT	A-17

CHAPTER 6

JOHN MUSA'S
LOGARITHMIC POISSON
EXECUTION TIME MODEL

6.1 INTRODUCTION

This chapter describes the SMFLIB routines associated with the Logarithmic Poisson model developed by John Musa (Reference 4). This model is based on the amount of Central Processing Unit (CPU) time involved in testing, as is Musa's other model (Chapter 5). The three (main level) SMFLIB routines that support this model are SMSAMD, SMSAMA, and SMSAPR.

The first SMFLIB routine, SMSAMD, is used to calculate the Maximum Likelihood (ML) estimates. The Dekker-Brent algorithm is used to solve the equations. (Refer to Reference 4 for the equations of those estimates and Reference 3 for information on the algorithm.)

The second SMFLIB routine, SMSAMA, is used to perform the four model applicability analyses (Reference 6) for the ML method of execution. This includes the analysis over accuracy, bias, trend, and noise. The accuracy of the model is calculated as:

$$ACC = \sum_{I=NSB}^{NSE} -\log(B_0) + \log(B_1) - \log(TMP1) + B_0 * (\log(TMP2) - \log(TMP1))$$

where TMP1 and TMP2 are defined as:

$$TMP1 = 1.0 + B_1 * \sum_{J=1}^{I+1} DAT_J \quad \quad \quad TMP2 = 1.0 + B_1 * \sum_{J=1}^I DAT_J$$

and NSB and NSE are the user-specified starting and ending points for the iteration. (These values are usually set to one-half the number of failures and the number of failures minus one, respectively.) For each iteration, the SMSAMD routine is executed using I failures; and the estimates for BETA0 (B_0) and BETA1 (B_1) based on those I failures are placed in the prior equation. The summations of the DAT(J)'s are the total amount of testing time that has been done up to the time of the current and next failure occurrences. DAT(J) is the time between the (J-1)st and Jth failure occurrences.

The vector for the bias of the model is calculated as:

$$U_I = 1.0 - \left(\frac{TMP2}{TMP1} \right)^{B_0}$$

as I goes from NSB to NSE (using TMP1 and TMP2 as defined for the accuracy). Once all iterations are done, the vector is sorted in ascending order; and the Kolmogorov distance is calculated as:

$$DP = \max\left(0.0, \left(\frac{I}{NSR} - U_I \right), \left(U_I - \frac{I-1}{NSR} \right)\right)$$

as I goes from 1 to NSR (the range formed from NSB to NSE).

The calculation for the trend of the model starts with the same function as described for the bias; however, the values are scaled as:

$$T_I = -\text{LOG}(1.0 - U_I)$$

as I goes from NSB to NSE and then scaled a second time as:

$$Y_I = \sum_{K=1}^I \frac{T_K}{\text{SUMTI}}$$

where SUMTI is the summation of the T(I)'s.

The calculation for the noise of the model is calculated as:

$$\text{NSE} = \sum_{I=\text{NSB}}^{\text{NSE}} \left| \frac{M_I - M_{I-1}}{M_{I-1}} \right|$$

where M is calculated as:

$$M_I = \frac{B_1 * \sum_{J=1}^I \text{DAT}_J + 1.0}{B_1 * 0.5^{\frac{1}{B_1}}} - \frac{1.0}{B_1} - \sum_{J=1}^I \text{DAT}_J$$

The third SMFLIB routine, SMSAPR, is used to calculate the expected number of failures for an additional testing time, to calculate the expected number of failures and the additional testing time to achieve a desired intensity function (expressed as failures per CPU time unit), and to generate a vector of predicted Time-Between-Failures (TBF) data. The Goodness-of-Fit of those predicted values to the original observed data is also computed during the generation of the predicted data vector. The estimate of the number of failures expected during an additional testing time is calculated as:

$$\text{PDAT}_1 = \text{LOG} \left(\frac{B_1 * \text{EXTM} + 1.0}{B_1 * \text{SUMTBF} + 1.0} \right)$$

where BETA1 is one of the model parameters, SUMTBF is the total amount of testing time that has been done up to the current failure occurrence, and EXTM is that time plus the additional amount.

The estimates of the predicted number of failures and the additional testing time to achieve a desired intensity function are calculated as:

$$\text{PDAT}_1 = B_0 * \text{LOG} \left(\frac{\text{CIF}}{\text{DIF}} \right)$$

and

$$\text{PDAT}_2 = B_0 * \left(\frac{1.0}{\text{DIF}} - \frac{1.0}{\text{CIF}} \right)$$

where BETA0 is as previously defined, CIF is the current intensity function, and DIF is the desired intensity function.

The estimate for the I^{th} failure (i.e., $I-1$ failures have been observed), where I ranges from one to NS , is calculated as:

$$P_{DAT_I} = \frac{1.0}{B_1} * \left(\left(\frac{B_0}{B_0 - 1.0} \right)^I - 1.0 \right) - \frac{1.0}{B_1} * \left(\left(\frac{B_0}{B_0 - 1.0} \right)^{I-1} - 1.0 \right)$$

where $BETA0$ and $BETA1$ are as previously defined.

The Goodness-of-Fit calculation involves the same equations as shown for the bias calculation; however, the range goes from 1 to NS , and the values for $BETA0$ and $BETA1$ are calculated utilizing the entire failure history.

6.2 ACCESS LINES AND ARGUMENT LISTS

6.2.1 SMSAMD Routine

The access line to the SMSAMD routine is as follows:

```
CALL SMSAMD (NS      ,DAT      ,SUMTBF ,BETA0  ,BETA1  ,CIF
              ,IIF      ,RFLAG   )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- NS** = Input integer containing the number of software failures observed to date.
- DAT** = Input real vector, of length greater than or equal to NS , containing the CPU TBF data (in any consistent form of elapsed time measurement).
- SUMTBF** = Input real containing the total amount of testing time, including any failure-free test time.
- BETA0** = Output real containing the $BETA0$ parameter of the model. This value should only be considered final when **RFLAG** has a value of zero.
- BETA1** = Output real containing the $BETA1$ parameter of the model. This value should only be considered final when **RFLAG** has a value of zero.
- CIF** = Output real containing the model estimate of the current intensity function. The value should only be considered final when **RFLAG** has a value of zero.
- IIF** = Output real containing the model estimate of the initial intensity function. The value should only be considered final when **RFLAG** has a value of zero.
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing and three indicates the data are not appropriate for the model.

6.2.2 SMSAMA Routine

The access line to the SMSAMA routine is as follows:

```
CALL SMSAMA (DAT      ,NS      ,NSB      ,NSE      ,NSR      ,TYP
             ,STAT     ,RFLAG   ,INDX     ,V        ,VPRE     )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- DAT** = Input real vector, of length greater than or equal to NS, containing the CPU TBF data (in any consistent form of elapsed time measurement).
- NS** = Input integer containing the number of software failures observed to date.
- NSB** = Input integer containing the starting point for the analysis.
- NSE** = Input integer containing the ending point for the analysis.
- NSR** = Input integer containing the range of the analysis.
- TYP** = Input integer flag indicating the type of analysis to be performed, where:
- a. One indicates the accuracy
 - b. Two indicates the bias
 - c. Three indicates the noise
 - d. Four indicates the trend
- STAT** = Output real containing the statistic for the analysis, where the following is returned (based on the assignment of the argument TYP):
- a. Computed accuracy
 - b. Computed Kolmogorov distance for the U-Plot
 - c. Computed noise
 - d. Computed Kolmogorov distance for the Y-Plot
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, and five indicates the model applicability analysis cannot be performed.
- INDX** = Output integer containing the number of iterations performed during the model applicability analysis.
- V** = Output real vector, of length greater than or equal to NSR, containing the data of the accuracy scatter-plot, bias U-Plot, or trend Y-Plot. The vector is not used for the noise analysis.

VPRE = Output real vector, of length greater than or equal to NSR, containing the data values for the bias U-Plot, prior to sorting. The vector is not used for the other three analysis types.

6.2.3 SMSAPR Routine

The access line to the SMSAPR routine is as follows:

```
CALL SMSAPR (NPV ,NS ,CIF ,DIF ,BETA0 ,BETA1
             ,EXTM ,DAT ,PDAT ,DVAL ,DFLG ,V )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- NPV** = Input integer containing the number of predicted values desired, where one indicates the prediction for the number of failures expected during an additional execution time, two indicates the expected number of failures and the corresponding test time required to achieve a specified intensity function, and (the value of) NS indicates the vector of predicted TBF data.
- NS** = Input integer containing the number of software failures observed to date. This argument is only used during the vector generation.
- CIF** = Input real containing the model estimate for the current intensity function (i.e., CIF from the SMSAMD access). This argument is only used during the predictions for the expected failure number and additional testing time (i.e., CIF is only used when NPV is set to two).
- DIF** = Input real containing the desired intensity function. This argument is only used during the predictions for the expected failure number and additional testing time (i.e., DIF is only used when NPV is set to two).
- BETA0** = Input real containing the model parameter, BETA0 (i.e., BETA0 from the SMSAMD access).
- BETA1** = Input real containing the model parameter, BETA1 (i.e., BETA1 from the SMSAMD access). This argument is not used during the predictions for the expected failure number and additional testing time (i.e., BETA1 is not used when NPV is set to two).
- EXTM** = Input real containing the desired execution time, when NPV is set to one. The input value should include the total amount of previous execution time (excluding any failure-free testing time). This argument is only used during the prediction for the number of failures expected during an additional execution time (i.e., EXTM is only used when NPV is set to one).
- DAT** = Input real vector, of length greater than or equal to NS, containing the CPU TBF data. This argument is used during the prediction for the number of failures

expected during an additional execution time and during the calculation of the Kolmogorov distance.

- PDAT** = Output real vector, of length greater than or equal to NPV, containing the predicted data value(s).
- DVAL** = Output real containing the Kolmogorov distance.
- DFLG** = Output integer flag indicating the significance of the computed Kolmogorov distance, where one indicates the model may not provide an adequate fit and two indicates the model may provide an adequate fit.
- V** = Scratch working real vector of length greater than or equal to NS.

6.3 EXAMPLES

The example executions of John Musa's Logarithmic Poisson model are contained on pages A-18 through A-20 and consist of an analysis of the TBF data set introduced in Section 1.4. (That data set consists of 30 data points, with the last point being the amount of failure-free execution time after the last failure happened.) Table 6-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

It can be seen that the number of failures is set to 29 (in the example) and the SUMTBF argument is set to the total time of testing (including the failure-free testing time). Had the final testing session terminated with an failure, NS would have been assigned the total number of data points (30) and SUMTBF would not have changed. Within the model, the failure-free test time is removed after the model parameters are computed. This brings the estimates related to execution time more in line with the other models (which do not have provisions for the failure-free test time).

TABLE 6-1. MUSA'S LOGARITHMIC POISSON MODEL EXAMPLE PAGES

MODEL APPLICABILITY ANALYSES	A-18
MODEL ESTIMATION EXECUTION	A-19
EXPECTED NUMBER OF FAILURES DURING ADDITIONAL TESTING	A-19
FAILURES AND TESTING TIME TO REACH A DESIRED INTENSITY FUNCTION	A-19
PREDICTED TBF VECTOR GENERATION	A-20

As previously mentioned, the desired execution time during the first prediction includes the total time of testing up through the last failure occurrence. Hence, an additional execution time of 1 hour (3600 seconds) results in EXTM being set to 4178.5 seconds (when the 578.5 seconds to the final failure is added).

CHAPTER 7

NON-HOMOGENEOUS POISSON MODEL
FOR EXECUTION TIME DATA

7.1 INTRODUCTION

This chapter describes the SMFLIB routines associated with the Non-homogeneous Poisson model proposed by Amrit Goel and Kazu Okumoto, and applied to execution time analysis (Reference 1). The three (main level) SMFLIB routines that support this model are SNPTMD, SNPTMA, and SNPTPR.

The first SMFLIB routine, SNPTMD, is used to calculate the estimates using the Maximum Likelihood (ML) method of execution. The equations used to derive the ML estimates are provided in Reference 1, page 4-77. The Dekker-Brent algorithm is used to find the solution to the pair of equations (Reference 3).

The second SMFLIB routine, SNPTMA, is used to perform the four model applicability analyses (Reference 6) for the ML method of execution. This includes the analysis over accuracy, bias, trend, and noise. The accuracy of the model is calculated as:

$$ACC = \sum_{I=NSB}^{NSE} -\log(TNOF) + \log(PCON) - TNOF \\ \cdot \left(\exp\left(-PCON \cdot \sum_{j=1}^I DAT_j\right) - \exp\left(-PCON \cdot \sum_{j=1}^{I+1} DAT_j\right) \right) - PCON \cdot \sum_{j=1}^{I+1} DAT_j$$

where NSB and NSE are the user-specified starting and ending points for the iteration. (These values are usually set to one-half the number of failures and the number of failures minus one, respectively.) For each iteration, the SNPTMD routine is executed using I failures; and the estimates for the total number of faults (TNOF) and the proportionality constant (PCON) based on those I failures are placed in the prior equation. The summations of the DAT(J)'s are the total amount of testing time that has been done up to the time of the current and next failure occurrences. DAT(J) is the time between the (J-1)st and Jth failure occurrences.

The vector for the bias of the model is calculated as:

$$U_I = 1.0 - \exp\left(-TNOF \cdot \left(\exp\left(-PCON \cdot \sum_{j=1}^I DAT_j\right) - \exp\left(-PCON \cdot \sum_{j=1}^{I+1} DAT_j\right) \right)\right)$$

as I goes from NSB to NSE. Once all iterations are done, the vector is sorted in ascending order; and the Kolmogorov distance is calculated as:

$$DP = \max\left(0.0, \left(\frac{I}{NSR} - U_I\right), \left(U_I - \frac{I-1}{NSR}\right)\right)$$

as I goes from 1 to NSR (the range formed from NSB to NSE).

The calculation for the trend of the model starts with the same function as described for the bias; however, the values are scaled as:

$$T_I = -\text{LOG}(1.0 - U_I)$$

as I goes from NSB to NSE and then scaled a second time as:

$$Y_I = \sum_{k=1}^I \frac{T_k}{\text{SUMTI}}$$

where SUMTI is the summation of the T(I)'s.

The calculation for the noise of the model is calculated as:

$$\text{NSE} = \sum_{I=\text{NSB}}^{\text{NSE}} \left| \frac{M_I - M_{I-1}}{M_{I-1}} \right|$$

where M is calculated as:

$$M_I = \frac{\text{LOG}(\text{TNOF}) - \text{PCON} \cdot \sum_{j=1}^I \text{DAT}_j - \text{LOG}\left(\text{LOG}(0.5) + \text{TNOF} \cdot \text{EXP}\left(-\text{PCON} \cdot \sum_{j=1}^I \text{DAT}_j\right)\right)}{\text{PCON}}$$

The third SMFLIB routine, SNPTPR, is used to calculate the prediction of the program reliability for a specified operational time, to calculate the prediction of the additional testing time required to achieve a specified reliability for a specified operational time, and to generate a vector of the predicted Time-Between-Failures (TBF) data. The Goodness-of-Fit of those predicted values to the original observed data is also computed during the generation of the predicted data vector. The estimate of the program's reliability for a specified operational time is calculated as:

$$\text{PDAT}_1 = \text{EXP}\left(-\text{TNOF} \cdot \left(\text{EXP}\left(-\text{PCON} \cdot \sum_{j=1}^{\text{NS}} \text{DAT}_j\right) - \text{EXP}\left(-\text{PCON} \cdot \left(\sum_{j=1}^{\text{NS}} \text{DAT}_j + \text{SOT}\right)\right)\right)\right)$$

where TNOF and PCON are estimates for the total number of faults and the proportionality constant of the model; SOT is the specified operational time; and the summation of the DAT(J)'s, where J ranges from one to NS, is the total amount of testing time that has been done up through the last failure occurrence.

The estimate of the required testing time to achieve a specified reliability for a specified operational time is calculated as:

$$\text{PDAT}_1 = \frac{\text{LOG}(\text{TNOF} \cdot (1.0 - \text{EXP}(-\text{PCON} \cdot \text{SOT}))) - \text{LOG}\left(\text{LOG}\left(\frac{1.0}{\text{SR}}\right)\right)}{\text{PCON}}$$

where TNOF, PCON, and SOT are as previously defined; and SR is the specified reliability.

The estimate for the I^{th} failure (i.e., $I-1$ failures have been observed), where I ranges from one to NS , is calculated as:

$$PDAT_I = \left(\frac{\sum_{j=1}^I DAT_j \cdot I}{TNOF \cdot \left(1.0 - \exp \left(-PCON \cdot \sum_{j=1}^I DAT_j \right) \right)} \right) - \left(\frac{\sum_{j=1}^{I-1} DAT_j \cdot I - 1}{TNOF \cdot \left(1.0 - \exp \left(-PCON \cdot \sum_{j=1}^{I-1} DAT_j \right) \right)} \right)$$

where $TNOF$ and $PCON$ are as previously defined, and the summations of the $DAT(J)$'s are the total amounts of testing time that have been done up through the previous ($I-1$) failure occurrences and the current (I).

The Goodness-of-Fit calculation involves the same equations as shown for the bias calculation; however, the range goes from 1 to NS , and the values for $TNOF$ and $PCON$ are calculated utilizing the entire failure history.

7.2 ACCESS LINES AND ARGUMENT LISTS

7.2.1 SNPTMD Routine

The access line to the SNPTMD routine is as follows:

```
CALL SNPTMD (DAT ,NS ,STATS ,RFLAG )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- DAT** = Input real vector, of length greater than or equal to NS , containing the Time-To-Failures (TTF) data (in any consistent form of elapsed time measurement). Unlike the other (execution time) SMFLIB routines, which all use TBF data, this routine uses TTF data.
- NS** = Input integer containing the number of software failures observed to date.
- STATS** = Output real vector, of length two, containing the model estimates for the proportionality constant of the model and the total number of faults. The values should only be considered final when **RFLAG** has a value of zero.
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, and four indicates the estimated number of faults is less than the number of faults found to date.

7.2.2 SNPTMA Routine

The access line to the SNPTMA routine is as follows:

```
CALL SNPTMA (DAT      ,NS      ,NSB      ,NSE      ,NSR      ,TYP
             ,STAT     ,RFLAG   ,INDX     ,V        ,VPRE     )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- DAT** = Input real vector, of length greater than or equal to NS, containing the TTF data (in any consistent form of elapsed time measurement).
- NS** = Input integer containing the number of software failures observed to date.
- NSB** = Input integer containing the starting point for the analysis.
- NSE** = Input integer containing the ending point for the analysis.
- NSR** = Input integer containing the range of the analysis.
- TYP** = Input integer flag indicating the type of analysis to be performed, where:
- a. One indicates the accuracy
 - b. Two indicates the bias
 - c. Three indicates the noise
 - d. Four indicates the trend
- STAT** = Output real containing the statistic for the analysis, where the following is returned (based on the assignment of the argument TYP):
- a. Computed accuracy
 - b. Computed Kolmogorov distance for the U-Plot
 - c. Computed noise
 - d. Computed Kolmogorov distance for the Y-Plot
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, four indicates the estimated number of faults is less than the number of faults found to date, and five indicates the model applicability analysis cannot be performed.
- INDX** = Output integer containing the number of iterations performed during the model applicability analysis.
- V** = Output real vector, of length greater than or equal to NSR, containing the data of the accuracy scatter-plot, bias U-Plot, or trend Y-Plot. The vector is not used for the noise analysis.

VPRE = Output real vector, of length greater than or equal to NSR, containing the data values for the bias U-Plot, prior to sorting. The vector is not used for the other three analysis types.

7.2.3 SNPTPR Routine

The access line to the SNPTPR routine is as follows:

```
CALL SNPTPR (NPV ,NS ,DAT ,PCON ,TNOF ,SOT
             ,SR ,PDAT ,DVAL ,DFLG ,V )
```

where the arguments of the call line, in the order of occurrence, are defined as:

NPV = Input integer containing the number of predicted values desired, where one indicates either the program reliability or the additional testing time (reference the description of the SR argument), and (the value of) NS indicates the vector of predicted TBF data.

NS = Input integer containing the number of software failures observed to date.

DAT = Input real vector, of length greater than or equal to NS, containing the TTF data (in any consistent form of elapsed time measurement).

PCON = Input real containing the model estimate for the proportionality constant [i.e., STATS(1) from the SNPTMD access].

TNOF = Input real containing the model estimate for the total number of faults [i.e., STATS(2) from the SNPTMD access].

SOT = Input real containing the specified operational time. This argument is not used during the vector generation.

SR = Input real containing the specified reliability during the prediction of the additional testing time required to achieve a specified reliability for a specified operational time. When NPV is set to one and SR is set to minus one, the program reliability for a specified operational time is calculated. This argument is not used during the vector generation.

PDAT = Output real vector, of length greater than or equal to NPV, containing the predicted data value(s).

DVAL = Output real containing the Kolmogorov distance.

DFLG = Output integer flag indicating the significance of the computed Kolmogorov distance, where one indicates the model may not provide an adequate fit and two indicates the model may provide an adequate fit.

V = Scratch working real vector of length greater than or equal to NS.

7.3 EXAMPLES

The example executions of the Non-homogeneous Poisson model (as applied to execution time analysis) are contained on pages A-21 and A-22 and consist of an analysis of the TBF data set introduced in Section 1.4. (That data set consists of 30 data points, with the last point being the amount of failure-free execution time after the last failure happened.) Table 7-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

The Non-homogeneous Poisson model accepts TTF data, not the standard TBF data. As the data described in Section 1.4 are recorded in the latter, the times were merged using the SMFTRN routine (Chapter 14). Additionally, it can be seen that the model does not have provisions for the final failure-free testing time. This time should be considered when the specified operational time is entered.

**TABLE 7-1. NON-HOMOGENEOUS POISSON MODEL (EXECUTION TIME)
EXAMPLE PAGES**

MODEL APPLICABILITY ANALYSES	A-21
MODEL ESTIMATION EXECUTION	A-22
PROGRAM RELIABILITY FOR SPECIFIED OPERATIONAL TIME	A-22
ADDITIONAL TESTING TIME TO REACH A SPECIFIED RELIABILITY FOR A SPECIFIED OPERATIONAL TIME	A-22
PREDICTED TBF VECTOR GENERATION	A-22

CHAPTER 8

**BROOKS AND MOTLEY'S
DISCRETE SOFTWARE RELIABILITY MODEL
FOR INTERVAL DATA**

8.1 INTRODUCTION

This chapter describes the SMFLIB routines associated with the Discrete Software Reliability model formulated by W. Brooks and R. Motley of the IBM Corporation (Reference 1). The three (main level) SMFLIB routines that support this model are SBAMMD, SBAMMA, and SBAMPR.

The first SMFLIB routine, SBAMMD, is used to calculate the estimates using the Binomial and Poisson methods of execution. Under both execution methods, the Trust Region algorithm is used. (Refer to Reference 1 for the model assumptions and the formulation, and Reference 3 for information on the algorithms and the starting values for the Trust Region.)

The second SMFLIB routine, SBAMMA, is used to perform the model applicability analysis over the accuracy (Reference 6). That value is calculated as follows for the Binomial and Poisson methods, respectively:

$$ACC = \sum_{I=NSB}^{NSE} -\log(TMP1)! - \log(CDAT_{I+1}) - \log(TMP1 - CDAT_{I+1})! \\ + CDAT_{I+1} * \log(TMP2) + (TMP1 - CDAT_{I+1}) * \log(1.0 - TMP2)$$

$$ACC = \sum_{I=NSB}^{NSE} -CDAT_{I+1} * \log(TMP1) + CDAT_{I+1} * \log(TMP2) \\ - TMP1 * TMP2 - \log(CDAT_{I+1})!$$

where TMP1 and TMP2 are defined as:

$$TMP1 = TNOF - PEC * MDAT_I \\ TMP2 = 1.0 - (1.0 - PED)^{LDAT_{I+1}}$$

and NSB and NSE are the user-specified starting and ending points for the iteration. (These values are usually set to one-half the number of intervals and the number of intervals minus one, respectively.) For each iteration, the SBAMMD routine is executed using I intervals; and the probability of correcting faults (PEC), the estimate for the total number of faults (TNOF), and the probability of observing faults (PED) based on those I intervals are placed in the prior equation. (PED is calculated using the equation shown in Paragraph 8.2.1.) The indexed elements of the CDAT, LDAT, and MDAT arrays contain the fault count, interval length, and cumulative fault observations, respectively. Note that the fractional portion of the program under test array (FDAT) is not used in the prior equations; testing of the complete program is assumed.

The third SMFLIB routine, SBAMPR, is used to calculate the prediction of the expected number of faults in the next testing interval, and to generate a vector of the predicted interval fault counts. The estimate of the expected number of faults in the next interval of testing is calculated as:

$$PDAT_1 = (1.0 - (1.0 - PED)^{EXL}) * (TNOF * EXF - PEC * ERM)$$

where TNOF and PED are estimates for the total number of faults and the probability of observing faults, PEC is the actual probability of correcting faults, EXL is the expected length of the next testing interval, and EXF and ERM are the expected fractional portion of code to be tested in the next interval and the total number of faults found to date in that fractional portion.

The estimate of the expected number of faults for the I^{th} interval, where I ranges from one to NS, is calculated as:

$$PDAT_I = (1.0 - (1.0 - PED)^{LDAT_I}) * (TNOF * FDAT_I - PEC * MDAT_I)$$

where TNOF, PED, and PEC are as previously defined; and the I^{th} elements of LDAT, FDAT, and MDAT replace the simple variables EXL, EXF, and ERM.

8.2 ACCESS LINES AND ARGUMENT LISTS

8.2.1 SBAMMD Routine

The access line to the SBAMMD routine is as follows:

```
CALL SBAMMD (ESF      ,MAXIC  ,NS      ,PEC      ,PED      ,TNOF
              ,FDAT     ,MDAT   ,CDAT   ,LDAT     ,STATS   ,COUNT
              ,RFLAG   )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- ESF = Input integer flag indicating the estimation selection type, where one indicates the Binomial method of estimation and two indicates the Poisson method.
- MAXIC = Input integer containing the maximum number of iterations to be performed, before a return to the calling program is made if convergence does not occur.
- NS = Input integer containing the number of testing intervals conducted to date.
- PEC = Input real containing the probability of correcting faults without inserting new ones. A value of one indicates that faults are never introduced during the correction process.
- PED = Input real containing the initial estimate (starting point to be used in the estimation process) for the probability of observing faults. A possible starting point may be calculated as:

$$PED = \frac{\sum_{I=1}^{NS} \left(\frac{CDAT_I}{\text{found faults}} \right) * LDAT_I * FDAT_I}{NS}$$

where I ranges from one to NS. However, other starting points (along with other initial estimates for TNOF) should be used to ensure the global maximum is reached.

TNOF = Input real containing the initial estimate (starting point to be used in the estimation process) for the total number of faults. A possible starting point may be calculated as:

$$TNOF = MAX \left(\frac{MDAT_I + CDAT_I}{FDAT_I} \right) + 1.0$$

where I ranges from one to NS. However, other starting points (along with other initial estimates for PED) should be used to ensure the global maximum is reached.

FDAT = Input real vector, of length greater than or equal to NS, containing the fractional portion of code being tested during each testing interval. A value of one indicates that the entire program was under test during the testing interval.

MDAT = Input real vector, of length greater than or equal to NS, containing the cumulative number of faults observed (in previous testing intervals) within the specified portion of code [i.e., If FDAT(I) is the fractional portion of code being tested in the Ith interval, MDAT(I) is the cumulative number of faults found in that part of the program on all previous testing occasions.]

CDAT = Input real vector, of length greater than or equal to NS, containing the interval fault counts. Although real storage locations are assigned for these data, the contents should be whole numbers.

LDAT = Input real vector, of length greater than or equal to NS, containing the interval testing lengths.

STATS = Output real vector, of length three, containing the estimation results. These values should only be considered final when RFLAG has a value of zero. The first two locations contain the model estimates for the total number of faults and the probability of detecting faults. The third location contains the (specified) probability of correcting faults without inserting new ones [i.e., the value passed to the routine in the argument PEC is simply returned in STATS(3)].

COUNT = Output integer containing the number of iterations performed to obtain the model estimates.

RFLAG = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, one indicates that the maximum iteration count was reached before convergence occurred, two indicates the Trust Region could not be adjusted properly (which may indicate other starting points

are required), and four indicates the estimated number of faults is less than the number of faults found to date.

8.2.2 SBAMMA Routine

The access line to the SBAMMA routine is as follows:

```
CALL SBAMMA (CDAT ,LDAT ,FDAT ,MDAT ,NS ,NSB
              ,NSE ,NSR ,BOPFLG ,MAXIC ,PEC ,STAT
              ,RFLAG ,INDX ,V )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- CDAT = Input real vector, of length greater than or equal to NS, containing the interval fault counts.
- LDAT = Input real vector, of length greater than or equal to NS, containing the interval testing lengths.
- FDAT = Input real vector, of length greater than or equal to NS, containing the fractional portion of code being tested during each testing interval.
- MDAT = Input real vector, of length greater than or equal to NS, containing the cumulative number of faults observed (in previous testing intervals) within the specified portion of code.
- NS = Input integer containing the number of testing intervals conducted to date.
- NSB = Input integer containing the starting point for the analysis.
- NSE = Input integer containing the ending point for the analysis.
- NSR = Input integer containing the range of the analysis.
- BOPFLG = Input integer flag indicating the estimation selection type, where one indicates the Binomial method of estimation and two indicates the Poisson method.
- MAXIC = Input integer containing the maximum number of iterations to be performed, before a return to the calling program is made if convergence does not occur.
- PEC = Input real containing the probability of correcting faults without inserting new ones.
- STAT = Output real containing the statistic for computed accuracy.
- RFLAG = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, one indicates that the maximum

iteration count was reached before convergence occurred, two indicates the Trust Region could not be adjusted properly (which may indicate other starting points are required), and four indicates the estimated number of faults is less than the number of faults found to date.

- INDX** = Output integer containing the number of iterations performed during the model applicability analysis.
- V** = Output real vector, of length greater than or equal to NSR, containing the data of the accuracy scatter-plot.

8.2.3 SBAMPR Routine

The access line to the SBAMPR routine is as follows:

```
CALL SBAMPR (NPV      ,NS      ,TNOF    ,PED      ,PEC      ,EXF
             ,EXL      ,ERM      ,FDAT    ,LDAT      ,MDAT      ,PDAT    )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- NPV** = Input integer containing the number of predicted values desired, where one indicates the number of faults expected in the next testing interval, and (the value of) NS indicates the vector of predicted fault counts for all conducted intervals.
- NS** = Input integer containing the number of testing intervals conducted to date.
- TNOF** = Input real containing the model estimate for the total number of faults [i.e., STATS(1) from the SBAMMD access].
- PED** = Input real containing the model estimate for the probability of observing faults [i.e., STATS(2) from the SBAMMD access].
- PEC** = Input real containing the (specified) probability of correcting faults without inserting new ones [i.e., PEC or STATS(3) from the SBAMMD access].
- EXF** = Input real containing the expected fractional portion of code to be tested in the next testing interval. This argument is not used during the vector generation.
- EXL** = Input real containing the expected length of the next testing interval. This argument is not used during the vector generation.
- ERM** = Input real containing the total number of faults found to date in the fractional portion of code to be tested in the next testing interval. This argument is not used during the vector generation.
- FDAT** = Input real vector, of length greater than or equal to NS, containing the fractional portion of code being tested during each testing interval. This argument is not

used during the prediction for the total number of faults expected in the next testing interval.

LDAT = Input real vector, of length greater than or equal to NS, containing the interval testing lengths. This argument is not used during the prediction for the total number of faults expected in the next testing interval.

MDAT = Input real vector, of length greater than or equal to NS, containing the cumulative number of faults observed (in previous testing intervals) within the specified portion of code. This argument is not used during the prediction for the total number of faults expected in the next testing interval.

PDAT = Output real vector, of length greater than or equal to NPV, containing the predicted data value(s).

8.3 EXAMPLES

The example executions of Brooks and Motley's Discrete Software Reliability model are contained on pages B-3 through B-6 and consist of two analyses of the interval data set introduced in Section 1.4; the first is for the Binomial and the second is for the Poisson. (That data set consists of 30 testing intervals, all with equal testing lengths of one.) Table 8-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

The Discrete Software Reliability model allows for the input of the fractional portion of a program that is under test, through the FDAT vector. For simplicity (in the examples), all fractions of FDAT are set to one, indicating that the entire program is always under test. This assignment, additionally, simplifies the assignment for the MDAT vector. Since the MDAT vector is the cumulative number of faults observed (in previous intervals) for the portion specified in FDAT, then MDAT(I) is simply the summation of the CDAT(J)'s, where J ranges from one to I-1.

It can be seen that the initial estimates for the total number of faults and the probability of detecting faults were simply set by the calculations shown in Paragraph 8.2.1. Had this been a real software reliability analysis, other starting points should have been explored to ensure that the global maximum had been reached.

Additionally, it can be seen that both the Binomial and Poisson methods resulted in an estimate for the total number of faults in the program of 305. Since 305 faults had already occurred, the prediction for the number of faults in the next interval of testing was not made for this data set.

TABLE 8-1. DISCRETE SOFTWARE RELIABILITY MODEL EXAMPLE PAGES

METHOD OF EXECUTION (ESF)	BINOMIAL	POISSON
MODEL APPLICABILITY ANALYSIS	B-3	B-5
MODEL ESTIMATION EXECUTIONS	B-3	B-5
EXPECTED FAULTS IN THE NEXT INTERVAL OF TESTING	N/A	N/A
EXPECTED FAULT COUNTS VECTOR GENERATIONS	B-4	B-6

CHAPTER 9

GENERALIZED POISSON MODEL
FOR INTERVAL DATA

9.1 INTRODUCTION

This chapter describes the SMFLIB routines associated with the Generalized Poisson model for interval data, presented by R. Schafer, J. Alter, J. Angus, and S. Emoto (Reference 1). The three (main level) SMFLIB routines that support this model are SGPOMD, SGPOMA, and SGPOPR.

The first SMFLIB routine, SGPOMD, is used to calculate the estimates using the Maximum Likelihood (ML) and Least Squares (LS) methods of execution. The routine, additionally, allows for three weighting functions to be applied to the interval testing lengths. The three functions are:

- a. Weighting the testing lengths by the power two and then dividing by two (Schick-Wolverton model)
- b. Weighting the testing lengths by the power ALPHA (where the value is input)
- c. Weighting the testing lengths by the power ALPHA (where the value is estimated)

The first and second weighting functions can be applied during both the ML and LS methods of execution. The third weighting function is only allowed during a ML estimation, in which the testing lengths are not equal for all intervals.

The Dekker-Brent algorithm is used to find the ML estimates (when ALPHA is not estimated) and the LS estimates. The Trust Region algorithm is used for the ML estimation when ALPHA is estimated. (Refer to Reference 1 for the equations of those estimates, and Reference 3 for information on the algorithms and the starting values for the Trust Region.)

The second SMFLIB routine, SGPOMA, is used to perform the model applicability analysis over the accuracy (Reference 6). The calculation can only be made for the ML ALPHA input method, and is calculated as:

$$ACC = \sum_{I=NSB}^{NSE} -CDAT_{I-1} * LOG(PCON) + CDAT_{I-1} * LOG(TNOF - MDAT_I) + CDAT_{I-1} * LOG(LDAT_{I-1}^A) - PCON * (TNOF - MDAT_I) * (LDAT_{I-1}^A) - LOG(CDAT_{I-1})$$

where NSB and NSE are the user-specified starting and ending points for the iteration. (These values are usually set to one-half the number of intervals and the number of intervals minus one, respectively.) For each iteration, the SGPOMD routine is executed using I intervals; and the user-specified values for ALPHA (A), the total number of faults (TNOF), and the proportionality constant (PCON) based on those

I intervals are placed in the prior equation. The indexed elements of the CDAT, LDAT, and MDAT arrays contain the fault counts, interval lengths, and cumulative fault corrections, respectively.

The third SMFLIB routine, SGPOPR, is used to calculate the prediction of the expected number of faults in the next testing interval and to generate a vector of the predicted interval fault counts. The estimate of the expected number of faults in the next interval of testing is calculated as:

$$PDAT_1 = PCON * (TNOF - (MDAT_{NS} + LCOR)) * EXL$$

where TNOF and PCON are estimates for the total number of faults and the proportionality constant of the model, MDAT(NS) is the number of faults that were corrected up through the end of last testing interval, LCOR is the number of faults that were corrected since the last interval, and EXL is the expected length of the next testing interval (weighted by the same function as used during the model execution). The estimates for the confidence interval about that estimate are calculated as:

$$PDAT_2 = PDAT_1 - 1.95 * \sqrt{VAR}$$

and

$$PDAT_3 = PDAT_1 + 1.95 * \sqrt{VAR}$$

where the formula for the variance (VAR) is provided in Reference 1, page 4-48.

The estimate of the expected number of faults for the Ith interval, where I ranges from one to NS, is calculated as:

$$PDAT_I = PCON * (TNOF - MDAT_I) * LDW_I$$

where PCON and TNOF are as previously defined, MDAT(I) is the number of faults that were corrected up through the end of the Ith testing interval, and LDW(I) is the associated length of the testing interval (weighted by the same function as used during the model execution).

9.2 ACCESS LINES AND ARGUMENT LISTS

9.2.1 SGPOMD Routine

The access line to the SGPOMD routine is as follows:

```
CALL SGPOMD (ESF      ,WFSF    ,ALPHA  ,AN      ,MAXIC  ,CDAT
             ,MDAT     ,NS      ,LDAT   ,STATS   ,COUNT ,RFLAG )
```

where the arguments of the call line, in the order of occurrence, are defined as:

ESF = Input integer flag indicating the estimation selection type, where one indicates the ML method of execution and two indicates the LS method.

WFSF = Input integer flag indicating the weighting to be applied to the LDAT vector, where one indicates the Schick-Wolverton, two indicates raising the values by the input ALPHA, and three indicates raising the values by an estimated ALPHA. The third weighting function is only allowed during the ML estimation, in which the testing lengths are not equal for all intervals.

- ALPHA** = Input real containing either the input or initial estimate (starting point to be used in the estimation process) for ALPHA. This argument is not used during executions using the Schick-Wolverton weighting function.
- AN** = Input real containing the initial estimate (starting point to be used in the estimation process) for the total number of faults. This argument is only used during ML executions, in which ALPHA is estimated. The current number of fault observations plus one may be used as a possible initial estimate; however, other starting points (along with other initial estimates for ALPHA) should be used to ensure the global maximum is reached.
- MAXIC** = Input integer containing the maximum number of iterations to be performed, before a return to the calling program is made if convergence does not occur. This argument is only used during ML executions, in which ALPHA is estimated.
- CDAT** = Input real vector, of length greater than or equal to NS, containing the interval fault counts. Although real storage locations are assigned for these data, the contents should be whole numbers.
- MDAT** = Input real vector, of length greater than or equal to NS, containing the cumulative number of fault corrections. [MDAT(I) is the cumulative number of faults corrected up through the end of the Ith testing interval.]
- NS** = Input integer containing the number of testing intervals conducted to date.
- LDAT** = On input, the real vector, of length greater than or equal to NS, containing the interval testing lengths. On output, the vector with the weighting function applied if WFSF is set to one or two. The vector is unchanged during the ML execution, in which ALPHA is estimated.
- STATS** = Output real array, of dimension four by three, containing the estimation results. These values should only be considered final when RFLAG has a value of zero. The first column of the STATS array contains the model estimates for:
- a. Proportionality constant
 - b. Total number of faults
 - c. Total number of faults remaining in the program
 - d. ALPHA (if ALPHA is estimated); otherwise, zero
- The second and third columns of the first three rows for the ML method contain the 95-percent confidence intervals for the associated estimates. The second and third columns are not used during the LS method and contain zeroes. The second and third columns of the fourth row are never used and contain zeroes.
- COUNT** = Output integer containing the number of iterations performed to obtain the model estimates.

RFLAG = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, one indicates that the maximum iteration count was reached before convergence occurred, two indicates the Trust Region could not be adjusted properly (which may indicate other starting points are required), three indicates the data are not appropriate for the model, and four indicates the estimated number of faults is less than the number of faults found to date. Values of zero, three, and four are associated to executions involving the Dekker-Brent algorithm; values of zero, one, two, and four are associated to the execution involving the Trust Region algorithm.

9.2.2 SGPOMA Routine

The access line to the SGPOMA routine is as follows:

```
CALL SGPOMA (CDAT ,LDAT ,MDAT ,NS ,NSB ,NSE
             ,NSR ,ALPHA ,STAT ,RFLAG ,INDX ,V
             ,SDAT )
```

where the arguments of the call line, in the order of occurrence, are defined as:

CDAT = Input real vector, of length greater than or equal to NS, containing the interval fault counts.

LDAT = Input real vector, of length greater than or equal to NS, containing the interval testing lengths.

MDAT = Input real vector, of length greater than or equal to NS, containing the cumulative number of fault corrections.

NS = Input integer containing the number of testing intervals conducted to date.

NSB = Input integer containing the starting point for the analysis.

NSE = Input integer containing the ending point for the analysis.

NSR = Input integer containing the range of the analysis.

ALPHA = Input real containing the input ALPHA for the weighting function.

STAT = Output real containing the statistic for computed accuracy.

RFLAG = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, and four indicates the estimated number of faults is less than the number of faults found to date.

- INDX** = Output integer containing the number of iterations performed during the model applicability analysis.
- V** = Output real vector, of length greater than or equal to NSR, containing the data of the accuracy scatter-plot.
- SDAT** = Scratch working real vector, of length greater than or equal to NS.

9.2.3 SGPOPR Routine

The access line to the SGPOPR routine is as follows:

```
CALL SGPOPR (NPV      ,NS      ,WFSF   ,CDAT   ,LDAT   ,MDAT
             ,EXL      ,LCOR    ,PCON   ,TNOF   ,ALPHA  ,PDAT   )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- NPV** = Input integer containing the number of predicted values desired, where one and three indicate the number of faults expected in the next testing interval, and (the value of) NS indicates the vector of predicted fault counts for all conducted intervals. The value one should be assigned for ML executions (in which ALPHA is estimated) and for both LS executions. The value three should be assigned for ML executions (in which ALPHA is not estimated), and results in the second and third locations of PDAT being assigned the estimates for the 95-percent confidence interval about that estimate.
- NS** = Input integer containing the number of testing intervals conducted to date.
- WFSF** = Input integer flag indicating the weighting which was applied to the LDAT vector (during the SGPOMD execution) where one indicates the Schick-Wolverton, two indicates raising the values by the input ALPHA, and three indicates raising the values by an estimated ALPHA. The third weighting function can only be allowed during the ML estimation, in which the testing lengths are not equal for all intervals.
- CDAT** = Input real vector, of length greater than or equal to NS, containing the interval fault counts. This argument is only used when the prediction is based upon the ML method and ALPHA is not estimated (i.e., CDAT is only used when NPV is set to three).
- LDAT** = Input real vector, of length greater than or equal to NS, containing the interval testing lengths. The values are not weighted, at this point. This argument is not used when the prediction is based upon the LS method or the ML method and ALPHA is estimated (i.e., LDAT is not used when NPV is set to one).
- MDAT** = Input real vector, of length greater than or equal to NS, containing the cumulative number of fault corrections.

- EXL** = Input real containing the expected length of the next testing interval. This argument is not used during the vector generation.
- LCOR** = Input real containing the number of fault corrections performed after the last testing interval. This argument is not used during the vector generation.
- PCON** = Input real containing the model estimate for the proportionality constant [i.e., STATS(1,1) from the SGPOMD access].
- TNOF** = Input real containing the model estimate for the total number of faults [i.e., STATS(2,1) from the SGPOMD access].
- ALPHA** = Input real containing either the input or estimated ALPHA value [i.e., ALPHA from the SGPOMD access during a weighting by an input value for ALPHA, or STATS(4,1) from the SGPOMD access during a weighting by an estimated value for ALPHA]. This argument is not used during executions using the Schick-Wolverton weighting function.
- PDAT** = Output real vector, of length greater than or equal to NPV, containing the predicted data value(s).

9.3 EXAMPLES

The example executions of the Generalized Poisson model are contained on pages B-7 through B-12 and consist of five analyses of the interval data set introduced in Section 1.4; the first three are for the ML treatment types 1, 2, and 3 executions and the last two are for the LS executions. Note, a treatment type 3 cannot be conducted for the LS method of execution. [That data set consists of 30 testing intervals, all with equal testing lengths of one. This example also uses the special assignment of variable lengths to satisfy the model requirement during the ML (ALPHA estimated) execution.] Table 9-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

The Generalized Poisson model allows for the specification of when faults are corrected through the MDAT vector. For simplicity, in the examples, all faults observed during the I^* interval are corrected at the beginning of the next testing interval. Since the MDAT vector is also cumulative, fault observations of one, two, and two for the first three intervals result in the assignments of zero, one, and three for the first three intervals.

The interpretation of the model estimates for the total number of faults and the total number of faults remaining in the program requires some additional explanation. The model "determined" that the total number of faults in the program was around 321 and that around 17 faults still remain in the program (from the ML execution with the Schick-Wolverton weighting, page B-7). The difference in those values is 304 faults. However, the summation of the elements of CDAT indicates that 305 faults have been observed. This is because the last fault (in location 30 of CDAT) has not been corrected (i.e., it is still in the program); hence, 17 faults still remain in the program.

Lastly, it can be seen that during the ML execution in which ALPHA was estimated (page B-10), the initial estimates for ALPHA and the total number of faults were simply set to 0 and 306. Had this been a real software reliability analysis, other starting points should have been explored to ensure that the global maximum had been reached.

TABLE 9-1. GENERALIZED POISSON MODEL EXAMPLE PAGES

METHOD OF EXECUTION (ESF)	ML			LS	
	1	2	3	1	2
WEIGHTING FUNCTION (WFSF)	1	2	3	1	2
MODEL APPLICABILITY ANALYSIS	N/A	B-8	N/A	N/A	N/A
MODEL ESTIMATION EXECUTIONS	B-7	B-8	B-10	B-11	B-12
EXPECTED FAULTS IN NEXT INTERVAL OF TESTING	B-7	B-8	B-10	B-11	B-12
EXPECTED FAULT COUNTS VECTOR GENERATIONS	B-7	B-9	B-10	B-11	B-12

CHAPTER 10

NON-HOMOGENEOUS POISSON MODEL
FOR INTERVAL DATA

10.1 INTRODUCTION

This chapter describes the SMFLIB routines associated with the Non-homogeneous Poisson model as applied to interval data analysis (Reference 1). The three (main level) SMFLIB routines that support this model are SNPIMD, SNPIMA, and SNPIPR.

The first SMFLIB routine, SNPIMD, is used to calculate the estimates using the Maximum Likelihood (ML) and Least Squares (LS) methods of execution. The Dekker-Brent algorithm is used to find both the ML and LS estimates. (Refer to Reference 1 for the equations of those estimates and Reference 3 for information on the algorithm.)

The second SMFLIB routine, SNPIMA, is used to perform the model applicability analysis over the accuracy (Reference 6). The calculation can only be made for the ML method and is calculated as:

$$ACC = \sum_{I=NSB}^{NSE} -CDAT_{I+1} * LOG(TNOF) + CDAT_{I+1} * LOG(TMP1 - TMP2) \\ + TNOF * (TMP2 - TMP1) - LOG(CDAT_{I+1})$$

where TMP1 and TMP2 are defined as:

$$TMP1 = EXP(-PCON * LDAT_I) \quad TMP2 = EXP(-PCON * LDAT_{I+1})$$

and NSB and NSE are the user-specified starting and ending points for the iteration. (These values are usually set to one-half the number of intervals and the number of intervals minus one, respectively.) For each iteration, the SNPIMD routine is executed using I intervals; and the estimates for the total number of faults (TNOF) and the proportionality constant (PCON) based on those I intervals are placed in the prior equation. The indexed elements of the CDAT and LDAT arrays contain the fault count and interval length, respectively.

The third SMFLIB routine, SNPIPR, is used to calculate the prediction of the expected number of faults in the next testing interval, and to generate a vector of the predicted interval fault counts. The estimate of the expected number of faults in the next interval of testing is calculated as:

$$PDAT_1 = TNOF * \left(EXP\left(-PCON * \sum_{J=1}^{NS} LDAT_J\right) - EXP(-PCON) * \left(\sum_{J=1}^{NS} LDAT_J + EXL \right) \right)$$

where TNOF and PCON are estimates for the total number of faults and the proportionality constant of the model, the summation of the LDAT(J)'s is the sum of all interval lengths performed to date, and EXL is the expected length of the next testing interval.

The estimate of the expected number of faults for the I^{th} interval, where I ranges from one to NS , is calculated as:

$$PDAT_I = TNOF * \left(EXP \left(-PCON * \sum_{j=1}^{I-1} LDAT_j \right) - EXP \left(-PCON * \sum_{j=1}^I LDAT_j \right) \right)$$

where $TNOF$ and $PCON$ are as previously defined, and the summations of the $LDAT(J)$'s are the sums of the interval lengths up through the previous $(I-1)$ intervals and the current (I) .

10.2 ACCESS LINES AND ARGUMENT LISTS

10.2.1 SNPIMD Routine

The access line to the SNPIMD routine is as follows:

```
CALL SNPIMD (ESF ,CDAT ,LDAT ,NS ,STATS ,RFLAG )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- ESF = Input integer flag indicating the estimation selection type, where one indicates the ML method of execution and two indicates the LS method.
- CDAT = Input real vector, of length greater than or equal to NS , containing the interval fault counts. Although real storage locations are assigned for these data, the contents should be whole numbers.
- LDAT = Input real vector, of length greater than or equal to NS , containing the cumulative interval testing lengths [i.e., if a sample size of five is specified and all testing lengths are of an equal length (or intensity) of one, then the first five locations of $LDAT$ are assigned one through five, respectively].
- NS = Input integer containing the number of testing intervals conducted to date.
- STATS = Output real array, of dimension two by three, containing the estimation results. These values should only be considered final when $RFLAG$ has a value of zero. The first column of the $STATS$ array contains the model estimates for:
 - a. Proportionality constant
 - b. Total number of faults

The second and third columns for the ML method contain the 95-percent confidence intervals for the associated estimates. The second and third columns are not used during the LS method and contain zeroes.

- RFLAG = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not

appropriate for the model, and four indicates the estimated number of faults is less than the number of faults found to date.

10.2.2 SNPIMA Routine

The access line to the SNPIMA routine is as follows:

```
CALL SNPIMA (CDAT ,LDAT ,NS ,NSB ,NSE ,NSR
             ,STAT ,RFLAG ,INDX ,V )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- CDAT = Input real vector, of length greater than or equal to NS, containing the interval fault counts.
- LDAT = Input real vector, of length greater than or equal to NS, containing the cumulative interval testing lengths.
- NS = Input integer containing the number of testing intervals conducted to date.
- NSB = Input integer containing the starting point for the analysis.
- NSE = Input integer containing the ending point for the analysis.
- NSR = Input integer containing the range of the analysis.
- STAT = Output real containing the statistic for computed accuracy.
- RFLAG = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, and four indicates the estimated number of faults is less than the number of faults found to date.
- INDX = Output integer containing the number of iterations performed during the model applicability analysis.
- V = Output real vector, of length greater than or equal to NSR, containing the data of the accuracy scatter-plot.

10.2.3 SNPIPR Routine

The access line to the SNPIPR routine is as follows:

```
CALL SNPIPR (NPV ,NS ,TNOF ,PCON ,LDAT ,EXL
             ,PDAT )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- NPV** = Input integer containing the number of predicted values desired, where one indicates the number of faults expected in the next testing interval, and (the value of) **NS** indicates the vector of predicted fault counts for all conducted intervals.
- NS** = Input integer containing the number of testing intervals conducted to date.
- TNOF** = Input real containing the model estimate for the total number of faults [i.e., **STATS(2,1)** from the **SNPIMD** access].
- PCON** = Input real containing the model estimate for the proportionality constant [i.e., **STATS(1,1)** from the **SNPIMD** access].
- LDAT** = Input real vector, of length greater than or equal to **NS**, containing the cumulative interval testing lengths.
- EXL** = Input real containing the expected length of the next testing interval. This argument is not used during the vector generation.
- PDAT** = Output real vector, of length greater than or equal to **NPV**, containing the predicted data value(s).

10.3 EXAMPLES

The example executions of the Non-homogeneous Poisson model (as applied to interval data analysis) are contained on pages B-13 and B-14 and consist of two analyses of the interval data set introduced in Section 1.4; the first is for the ML and the second is for the LS. (That data set consists of 30 testing intervals, all with equal testing lengths of one.) Table 10-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

**TABLE 10-1. NON-HOMOGENEOUS POISSON MODEL (INTERVAL DATA)
EXAMPLE PAGES**

METHOD OF EXECUTION (ESF)	ML	LS
MODEL APPLICABILITY ANALYSIS	B-13	N/A
MODEL ESTIMATION EXECUTIONS	B-13	B-14
EXPECTED FAULTS IN NEXT INTERVAL OF TESTING	B-13	B-14
EXPECTED FAULT COUNTS VECTOR GENERATIONS	B-13	B-14

The Non-homogeneous Poisson model accepts the cumulative testing lengths, not the lengths per testing interval. As the data described in Section 1.4 are recorded in the latter, the interval lengths (of LDAT) were merged using the SMFTRN routine (Chapter 14).

CHAPTER 11

**SCHNEIDEWIND'S
MAXIMUM LIKELIHOOD MODEL
FOR INTERVAL DATA**

11.1 INTRODUCTION

This chapter describes the SMFLIB routines associated with Schneidewind's Maximum Likelihood model (Reference 1). The three (main level) SMFLIB routines that support this model are SSDWMD, SSDWMA, and SSDWPR.

The first SMFLIB routine, SSDWMD, is used to calculate the estimates for Schneidewind's model. This model allows for three different possible treatment types which can be used to model differences between the earlier testing intervals and current testing intervals (Reference 1, pages 4-66 through 4-71). The three treatment types allow for:

- a. Using the fault counts from all testing intervals individually
- b. Ignoring the fault counts from the first S-1 testing intervals, using only the fault counts from the specified (S) testing interval through the total number of testing intervals
- c. Combining the fault counts from the first through S-1 testing intervals, using that combined value as the first point, and using the remaining fault counts individually

The Dekker-Brent algorithm is used to find the solution to the equations for this model. (Refer to Reference 3 for information on the algorithm.)

The Schneidewind model (for this revision to the SMFLIB) has been enhanced to include a modification to the equation for the weighted sums-of-squares between the predicted and observed fault counts, the addition of the equation for the corresponding mean square error based on the predicted number of cumulative failure counts and the actual number of cumulative failure counts (MSE_p), and the mean square error for time to the next failure (MSE_T). This brings the library more in line with Schneidewind's current research in the area of the selection of the optimum starting point (Reference 7).

The second SMFLIB routine, SSDWMA, is used to perform the model applicability analysis over the accuracy (Reference 6). The calculation can only be made for the treatment type 1 method (which uses all intervals individually) and is calculated as:

$$ACC = \sum_{I=NSB}^{NSE} -CDAT_{I+1} * LOG(TMP) - TMP - LOG(CDAT_{I+1})$$

where TMP is defined as:

$$TMP = \frac{A}{B} * (EXP(-B * I) - EXP(-B * (I+1)))$$

and NSB and NSE are the user-specified starting and ending points for the iteration. (These values are usually set to one-half the number of intervals and the number of intervals minus one, respectively.) For each iteration, the SSDWMD routine is executed using I intervals; and the model parameters, ALPHA (A) and BETA (B), based on those I intervals are placed in the prior equation. The indexed element of the CDAT array contains the fault count.

The third SMFLIB routine, SSDWPR, is used to calculate the prediction of the expected number of faults in the next testing interval, to calculate the prediction of the number of testing intervals to detect a desired number of faults, and to generate a vector of the predicted interval fault counts (Reference 7). The number of assigned elements in the returned vector is the result of both the number of intervals performed to date and the treatment type. For treatment type one, the vector will contain a prediction for each interval. For treatment type two, the vector will be reduced by S-1 predictions; therefore, to re-instate continuity to the observed fault counts (as would be needed to perform any type of Goodness-of-Fit analysis or plot), the first S-1 intervals should be eliminated from the observed data vector. For treatment type three, the vector will be reduced by S-2 predictions. To re-instate continuity here, the first S-1 intervals (of observed fault counts) should be summed on interval one, and intervals two through S-1 should be eliminated from the observed data vector.

11.2 ACCESS LINES AND ARGUMENT LISTS

11.2.1 SSDWMD Routine

The access line to the SSDWMD routine is as follows:

```
CALL SSDWMD (CDAT ,NS ,TSF ,TSS ,STATS ,RFLAG )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- CDAT = Input real vector, of length greater than or equal to NS, containing the interval fault counts. Although real storage locations are assigned for these data, the contents should be whole numbers.
- NS = Input integer containing the number of testing intervals conducted to date.
- TSF = Input integer flag indicating the treatment selection type, where one indicates using all testing intervals individually, two indicates ignoring the first through TSS-1 testing intervals, and three indicates combining the first through TSS-1 testing intervals.
- TSS = Input integer containing the starting point for the treatment selection. This argument must be set to one for treatments considering all testing intervals (i.e., TSS must be set to one if TSF is set to one).

- STATS** = Output real vector, of length six, containing the model estimates for the parameters **BETA** and **ALPHA**, the model estimate for the total number of faults, the weighted sums-of-squares between the predicted and observed fault counts, the corresponding MSE_r , and the corresponding MSE_t . The total number of faults estimate is calculated from $ALPHA/BETA$; hence, the location is returned with a -1.0 to indicate that a zero **BETA** stopped the calculation. Similarly, if the calculation for the MSE_t is terminated by the attempt to take the log of a non-positive number, the sixth element of the **STATS** vector is returned as -1.0. Additionally, the values should only be considered final when **RFLAG** has a value of zero.
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, and four indicates the estimated number of faults is less than the number of faults found to date.

11.2.2 SSDWMA Routine

The access line to the SSDWMA routine is as follows:

```
CALL SSDWMA (CDAT ,NS ,NSB ,NSE ,NSR ,STAT
              ,RFLAG ,INDX ,V )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- CDAT** = Input real vector, of length greater than or equal to **NS**, containing the interval fault counts.
- NS** = Input integer containing the number of testing intervals conducted to date.
- NSB** = Input integer containing the starting point for the analysis.
- NSE** = Input integer containing the ending point for the analysis.
- NSR** = Input integer containing the range of the analysis.
- STAT** = Output real containing the statistic for computed accuracy.
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, and four indicates the estimated number of faults is less than the number of faults found to date.
- INDX** = Output integer containing the number of iterations performed during the model applicability analysis.

V = Output real vector, of length greater than or equal to NSR, containing the data of the accuracy scatter-plot.

11.2.3 SSDWPR Routine

The access line to the SSDWPR routine is as follows:

```
CALL SSDWPR (NPV ,NS ,DNOF ,DNOP ,BETA ,ALPHA
             ,CDAT ,TSF ,TSS ,PDAT ,RFLAG )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- NPV** = Input integer containing the number of predicted values desired, where one indicates either the number of faults expected in the next DNOP testing intervals or the number of testing intervals required to detect a desired number of faults (reference the description of the DNOF argument), and (the value of) NS indicates the vector of predicted fault counts for the number of intervals used by the model (see Section 11.1).
- NS** = Input integer containing the number of testing intervals conducted to date.
- DNOF** = Input real containing the desired number of faults to observe. When NPV is set to one and DNOF is set to minus one, the expected number of faults in the next DNOP testing intervals is calculated. This argument is not used during the vector generation.
- DNOP** = Input real containing the desired number of periods to examine during the prediction of the expected number of faults. This argument is only used during that prediction.
- BETA** = Input real containing the model parameter, BETA [i.e., STATS(1) from the SSDWMD access].
- ALPHA** = Input real containing the model parameter, ALPHA [i.e., STATS(2) from the SSDWMD access].
- CDAT** = Input real vector, of length greater than or equal to NS, containing the interval fault counts.
- TSF** = Input integer flag indicating the treatment selection type, where one indicates using all testing intervals individually, two indicates ignoring the first through TSS-1 testing intervals, and three indicates combining the first through TSS-1 testing intervals.
- TSS** = Input integer containing the starting point for the treatment selection. This argument must be set to one for treatments considering all testing intervals (i.e., TSS must be set to one if TSF is set to one).

- PDAT** = Output real vector, of length greater than or equal to NPV, containing the predicted data value(s).
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing and one indicates the log of a non-positive number terminated the calculation for the number of testing intervals required to detect a specified number of faults. This argument is not used during the prediction for the number of faults in the next testing interval or during the vector generation.

11.3 EXAMPLES

The example executions of Schneidewind's model are contained on pages B-15 through B-18 and consist of three analyses of the interval data set introduced in Section 1.4; each example reflecting one of the treatment types. (That data set consists of 30 testing intervals, all with equal testing lengths of one.) Table 11-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

The executions pertaining to the second and third treatment types are for an example only; there is no intent to indicate that the assignments for the argument TSS are actually logical or meaningful.

Additionally, it can be seen that the second treatment type resulted in an estimate for the total number of faults in the program of 279. Since 305 faults had already occurred, the model fit is determined to be invalid and no future predictions are made for that treatment type for this data set.

TABLE 11-1. SCHNEIDEWIND'S MAXIMUM LIKELIHOOD MODEL EXAMPLE PAGES

TREATMENT TYPE (TSF)	TYPE 1	TYPE 2	TYPE 3
MODEL APPLICABILITY ANALYSIS	B-15	N/A	N/A
MODEL ESTIMATION EXECUTIONS	B-15	B-17	B-18
EXPECTED FAULTS IN NEXT INTERVAL OF TESTING	B-15	N/A	B-18
EXPECTED INTERVALS TO DETECT A DESIRED NUMBER OF FAULTS	B-15	N/A	B-18
EXPECTED FAULT COUNTS VECTOR GENERATIONS	B-16	N/A	B-18

CHAPTER 12

**YAMADA'S
S-SHAPED RELIABILITY GROWTH MODEL
FOR INTERVAL DATA**

12.1 INTRODUCTION

This chapter describes the SMFLIB routines associated with the S-Shaped Reliability Growth model proposed by Shigera Yamada, Mitsuru Ohba, and Shunji Osaki (Reference 5). Summarizing the abstract of that reference, this model is a modification of the Non-homogeneous Poisson model proposed by Amrit Goel and Kazu Okumoto, which is enhanced to model learning curves. The three (main level) SMFLIB routines that support this model are SESHMD, SESHMA, and SESHPR.

The first SMFLIB routine, SESHMD, is used to calculate the estimates. The Dekker-Brent algorithm is used to find the estimates. (Refer to Reference 5 for the equations of those estimates and Reference 3 for information on the algorithm.)

The second SMFLIB routine, SESHMA, is used to perform the model applicability analysis over the accuracy (Reference 6). The value is calculated as:

$$ACC = \sum_{I=NSB}^{NSE} -CDAT_{I+1} * LOG(TMP) - TMP - LOG(CDAT_{I+1})$$

where TMP is defined as:

$$TMP = TNOF * ((1.0 + PCON * LDAT_I) * EXP(-PCON * LDAT_I) - (1.0 + PCON * LDAT_{I+1}) * EXP(-PCON * LDAT_{I+1}))$$

and NSB and NSE are the user-specified starting and ending points for the iteration. (These values are usually set to one-half the number of intervals and the number of intervals minus one, respectively.) For each iteration, the SESHMD routine is executed using I intervals; and the estimates for the total number of faults (TNOF) and the proportionality constant (PCON) based on those I intervals are placed in the prior equation. The indexed elements of the CDAT and LDAT arrays contain the fault count and interval length, respectively.

The third SMFLIB routine, SESHPR, is used to calculate the prediction of the expected number of faults in the next testing interval, to calculate the probability that the software operates without fault during the next testing interval, and to generate a vector of the predicted fault counts. The estimate of the expected number of faults in the next interval of testing is calculated as:

$$PDAT_1 = TNOF * ((1.0 + PCON * LDAT_{NS}) * EXP(-PCON * LDAT_{NS}) - (1.0 + PCON * (LDAT_{NS} + EXL)) * EXP(-PCON * (LDAT_{NS} + EXL)))$$

where TNOF and PCON are as previously defined, LDAT(NS) is the sum of all interval lengths to date, and EXL is the expected length of the next testing interval.

The estimate for the reliability in the next testing interval is calculated as:

$$PDAT_1 = EXP(-PDAT_1)$$

where $PDAT(1)$ is the calculation for the previous prediction (i.e., the number of faults expected in the next testing interval).

The estimate of the expected number of faults for the I^{th} interval, where I ranges from one to NS , is calculated as:

$$PDAT_I = TNOF * ((1.0 + PCON * LDAT_{I-1}) * EXP(-PCON * LDAT_{I-1}) - (1.0 + PCON * LDAT_I) * EXP(-PCON * LDAT_I))$$

where $TNOF$ and $PCON$ are as previously defined, and the $LDAT(I)$'s are the sums of the interval lengths up through the previous $(I-1)$ and current (I) intervals.

12.2 ACCESS LINES AND ARGUMENT LISTS

12.2.1 SESHMD Routine

The access line to the SESHMD routine is as follows:

CALL SESHMD (CDAT ,LDAT ,NS ,STATS ,RFLAG)

where the arguments of the call line, in the order of occurrence, are defined as:

CDAT = Input real vector, of length greater than or equal to NS , containing the interval fault counts. Although real storage locations are assigned for these data, the contents should be whole numbers.

LDAT = Input real vector, of length greater than or equal to NS , containing the cumulative interval testing lengths [i.e., if a sample size of five is specified and all testing lengths are of an equal length (or intensity) of one, then the first five locations of **LDAT** are assigned one through five, respectively.]

NS = Input integer containing the number of testing intervals conducted to date.

STATS = Output real array, of dimension two by three, containing the estimation results. These values should only be considered final when **RFLAG** has a value of zero. The first column of the **STATS** array contains the model estimates for:

- a. Proportionality constant
- b. Total number of faults

The second and third columns contain the 95-percent confidence intervals for the associated estimates.

RFLAG = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not

appropriate for the model, and four indicates the estimated number of faults is less than the number of faults found to date.

12.2.2 SESHMA Routine

The access line to the SESHMA routine is as follows:

```
CALL SESHMA (CDAT ,LDAT ,NS ,NSB ,NSE ,NSR
             ,STAT ,RFLAG ,INDX ,V )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- CDAT = Input real vector, of length greater than or equal to NS, containing the interval fault counts.
- LDAT = Input real vector, of length greater than or equal to NS, containing the cumulative interval testing lengths.
- NS = Input integer containing the number of testing intervals conducted to date.
- NSB = Input integer containing the starting point for the analysis.
- NSE = Input integer containing the ending point for the analysis.
- NSR = Input integer containing the range of the analysis.
- STAT = Output real containing the statistic for computed accuracy.
- RFLAG = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, three indicates the data are not appropriate for the model, and four indicates the estimated number of faults is less than the number of faults found to date.
- INDX = Output integer containing the number of iterations performed during the model applicability analysis.
- V = Output real vector, of length greater than or equal to NSR, containing the data of the accuracy scatter-plot.

12.2.3 SESHPR Routine

The access line to the SESHPR routine is as follows:

```
CALL SESHPR (NPV ,NS ,TNOF ,PCON ,LDAT ,EXL
            ,PDAT )
```


where the arguments of the call line, in the order of occurrence, are defined as:

- NPV** = Input integer containing the number of predicted values desired, where one indicates the number of faults expected in the next testing interval, two indicates the probability that the software operates without fault during the next testing interval, and (the value of) NS indicates the vector of predicted fault counts for all conducted intervals.
- NS** = Input integer containing the number of testing intervals conducted to date.
- TNOF** = Input real containing the model estimate for the total number of faults [i.e., STATS(2,1) from the SESHMD access].
- PCON** = Input real containing the model estimate for the proportionality constant [i.e., STATS(1,1) from the SESHMD access].
- LDAT** = Input real vector, of length greater than or equal to NS, containing the cumulative interval testing lengths.
- EXL** = Input real containing the expected length of the next testing interval. This argument is not used during the vector generation.
- PDAT** = Output real vector, of length greater than or equal to NPV, containing the predicted data value(s).

12.3 EXAMPLES

The example executions of the S-Shaped Reliability Growth model are contained on pages B-19 and B-20 and consist of an analysis of the interval data set introduced in Section 1.4. (That data set consists of 30 testing intervals, all with equal testing lengths of one.) Table 12-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

**TABLE 12-1. YAMADA'S S-SHAPED RELIABILITY GROWTH MODEL
EXAMPLE PAGES**

MODEL APPLICABILITY ANALYSIS	B-19
MODEL ESTIMATION EXECUTION	B-19
EXPECTED FAULTS IN NEXT INTERVAL OF TESTING	B-19
PROBABILITY THAT THE SOFTWARE WILL OPERATE WITHOUT FAULT DURING THE NEXT TESTING INTERVAL	B-19
EXPECTED FAULT COUNTS VECTOR GENERATION	B-20

The S-Shaped Reliability Growth model accepts the cumulative testing lengths, not the lengths per testing interval. As the data described in Section 1.4 are recorded in the latter, the interval lengths (of LDAT) were merged using the SMFTRN routine (Chapter 14).

CHAPTER 13

UTILITY ROUTINE FOR DATA EDITS

13.1 INTRODUCTION

This chapter describes the SMFLIB utility routine, SMFEDT, which is used to edit a data set. The routine allows for four types of data editing over one or two data vectors. The different edit types are:

- a. Changing a specified element to a new value
- b. Deleting one or more adjacent elements
- c. Inserting one or more adjacent elements
- d. Combining two or more adjacent elements

13.2 ACCESS LINE AND ARGUMENT LIST

13.2.1 SMFEDT Routine

The access line to the SMFEDT routine is as follows:

```
CALL SMFEDT (PRC      ,ADA      ,ASZ      ,DSZ      ,DT1      ,DT2
             ,USZ      ,ERRFLG)
```

where the arguments of the call line, in the order of occurrence, are defined as:

- PRC = Input integer vector, of length four, containing the processing instructions for the data edit. The elements of PRC are defined as:
- PRC(1) - The edit type to be performed, where:
- a. One indicates change a specified element
 - b. Two indicates delete one or more adjacent elements
 - c. Three indicates insert one or more adjacent elements
 - d. Four indicates combine two or more adjacent elements
- PRC(2) - The number of data vectors to be edited (one or two).
- PRC(3) - If PRC(1) is equal to one, the index location of DT1 (DT2) to change.
- If PRC(1) is equal to two, the starting index location of DT1 (DT2) to delete.
- If PRC(1) is equal to three, the index location of DT1 (DT2) just prior to the insert.

- If PRC(1) is equal to four, the starting index location of DT1 (DT2) to merge.
 - PRC(4) - If PRC(1) is equal to one, this element is not used.
 - If PRC(1) is equal to two, the ending index location of DT1 (DT2) to delete.
 - If PRC(1) is equal to three, the number of elements to be inserted.
 - If PRC(1) is equal to four, the ending index location of DT1 (DT2) to merge.
- ADA** = Input real array, of dimension two by ASZ, containing the new data for the data vector(s).
 If PRC(1) is equal to one, ADA contains (in the first column of the first row) the value to be placed in the PRC(3) element of the DT1 vector. [If PRC(2) is equal to two, the value (in the first column of the of the second row) is to be placed in the PRC(3) element of the DT2 vector.] The remaining elements of ADA are not used.
 If PRC(1) is equal to two, this argument is not used.
 If PRC(1) is equal to three, the first PRC(4) columns of the first row of ADA are inserted after the PRC(3) element of the DT1 vector. [If PRC(2) is equal to two, the second row of ADA is similarly inserted in the DT2 vector.] The remaining elements of ADA are not used.
 If PRC(1) is equal to four, this argument is not used.
- ASZ** = Input integer containing the number of columns in the ADA array as dimensioned in the accessing program.
- DSZ** = Input integer containing the length of the DT1 and DT2 vectors as dimensioned in the accessing program.
- DT1** = On input, the real vector, of length DSZ, to be edited. On output, the edited vector. The vector is not altered if ERRFLG has a value other than zero.
- DT2** = On input, the real vector, of length DSZ, to be optionally edited. On output, the edited vector [if PRC(2) is equal to two]. The vector is not altered if ERRFLG has a value other than zero.
- USZ** = On input, the integer containing the number of used (assigned) elements in the DT1 and optional DT2 vectors. On output, the number used after the data edit was performed. The argument is not altered if ERRFLG has a value other than zero.
- ERRFLG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, one indicates the requested edit location(s) was outside of the bounds established by USZ, and two indicates the requested insertion would exceed the bounds of DSZ.

13.3 EXAMPLES

The example executions for various data edits are contained on pages C-3 through C-6 and consist of two modifications of the Time-Between-Failures (TBF) data set and two modifications of the interval data set (both introduced in Section 1.4). Table 13-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

TABLE 13-1. DATA EDIT EXAMPLE PAGES

CHANGE ELEMENT 21 FROM 30 TO 31 IN THE TBF DATA SET	C-3
DELETE ELEMENTS 10 THROUGH 15 IN THE TBF DATA SET	C-4
INSERT THREE NEW INTERVALS IN THE INTERVAL DATA SET	C-5
COMBINE PERIODS 10 THROUGH 13 IN THE INTERVAL DATA SET	C-6

CHAPTER 14

UTILITY ROUTINE FOR DATA TRANSFORMATIONS

14.1 INTRODUCTION

This chapter describes the SMFLIB utility routine, SMFTRN, which is used to transform a data set. The routine allows for six types of data transformations to be performed on the passed data vector. The different transformations that can be performed on a data vector (DAT), as I ranges from one to the data size, include the following:

- a. $\text{DAT}(I) = \text{LOG}(A * \text{DAT}(I) + B)$
- b. $\text{DAT}(I) = \text{EXP}(A * \text{DAT}(I) + B)$
- c. $\text{DAT}(I) = \text{DAT}(I)^A$
- d. $\text{DAT}(I) = \text{DAT}(I) + A$
- e. $\text{DAT}(I) = \text{DAT}(I) * A$
- f. $\text{DAT}(I) = \text{DAT}(I) + \text{DAT}(I-1)$

where the values for A and B are passed to the routine through the call line of the SMFTRN routine. The sixth transformation type provides an easy method to set each element of the data vector to the summation of the previous elements. [Note: DAT(I-1) on implementation becomes the summation of the previous I-1 elements.] This type of processing is necessary for models that require Time-To-Failures (TTF) data rather than Time-Between-Failures (TBF) data, and for models that require the cumulative testing lengths rather than the lengths per testing interval.

14.2 ACCESS LINE AND ARGUMENT LIST

14.2.1 SMFTRN Routine

The access line to the SMFTRN routine is as follows:

```
CALL SMFTRN (NS      ,A      ,B      ,TYPE  ,DAT  ,ERRFLG )
```

where the arguments of the call line, in the order of occurrence, are defined as:

NS = Input integer containing the number of elements to be transformed, which must be less than or equal to the dimension of the DAT vector in the accessing program.

- A** = Input real containing the first transformation variable (reference the description of the TYPE argument).
- B** = Input real containing the (optional) second transformation variable (reference the description of the TYPE argument).
- TYPE** = Input integer indicating the type of transformation to be performed, where:
- a. One indicates $\text{DAT}(I) = \text{LOG}(A * \text{DAT}(I) + B)$
 - b. Two indicates $\text{DAT}(I) = \text{EXP}(A * \text{DAT}(I) + B)$
 - c. Three indicates $\text{DAT}(I) = \text{DAT}(I)^A$
 - d. Four indicates $\text{DAT}(I) = \text{DAT}(I) + A$
 - e. Five indicates $\text{DAT}(I) = \text{DAT}(I) * A$
 - f. Six indicates $\text{DAT}(I) = \text{DAT}(I) + \text{DAT}(I-1)$
- DAT** = On input, the real vector, of length greater than or equal to NS, containing the data to be transformed. On output, the transformed vector. The vector is not altered if ERRFLG has a value other than zero.
- ERRFLG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing, one indicates the requested "TYPE=3" transformation could not be performed because of a negative DAT(I) value with a scale factor containing a decimal portion, and two indicates the requested "TYPE=1" transformation could not be performed because of a non-positive $(A * \text{DAT}(I) + B)$ value.

14.3 EXAMPLES

The example executions for various data transformations are contained on pages C-7 and C-8 and consist of four transformations of the TBF data set introduced in Section 1.4. Table 14-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

TABLE 14-1. DATA TRANSFORMATION EXAMPLE PAGES

CHANGE TO "LOG(10.0 * DAT(I) + 5.0)" (NATURAL LOG)	C-7
CHANGE TO "EXP(0.5 * DAT(I) + 1.0)"	C-7
CHANGE EXECUTION SECONDS TO EXECUTION MINUTES	C-8
CHANGE TBF DATA TO TTF DATA	C-8

CHAPTER 15

UTILITY ROUTINE FOR DATA STATISTICS

15.1 INTRODUCTION

This chapter describes the SMFLIB utility routine, SMFDST, which is used to generate summary statistics from the data vector. The resulting statistics include:

- a. Sum
- b. Median
- c. Lower Hinge
- d. Upper Hinge
- e. Minimum
- f. Maximum
- g. Mean
- h. Standard Deviation
- i. Sample Variance
- j. Skewness
- k. Kurtosis

where the sum is simply the summation of all assigned elements. The median is a measure of central tendency, such that 50 percent of the data have values below the median and 50 percent have values greater than the median.

The lower and upper hinges provide a measure of the spread of the data. The values are obtained such that the two sections of the data, determined by the median, are broken into two equal parts. Thus, 25 percent of the data are smaller than the lower hinge (75 percent being larger), and 75 percent of the data are smaller than the upper hinge (25 percent being larger).

The minimum and maximum are simply the smallest and largest values in the data set, and the mean is simply the average of the data. The standard deviation and the variance are calculated based on the sample size minus one.

The skewness is a measure of the symmetry of the sample and the kurtosis indicates how "peaked" the sample is.

15.2 ACCESS LINE AND ARGUMENT LIST

15.2.1 SMFDST Routine

The access line to the SMFDST routine is as follows:

```
CALL SMFDST (NS      ,NST      ,DAT      ,STATS  )
```

where the arguments of the call line, in the order of occurrence, are defined as:

- NS** = Input integer containing the number of elements over which the statistics are to be obtained.
- NST** = Input integer containing the number of statistics desired, where the only allowed values are 6 and 11. (Reference the description of the STATS argument to see the effect.) This assignment is governed, to some extent, by the type of data values present. For example, an interval testing length vector containing all equal lengths could only be executed for 6 values; attempting 11 causes an execution error.
- DAT** = On input, the real vector, of length greater than or equal to NS, from which the statistics are derived. On output, sorted in ascending order.
- STATS** = Output real vector, of length NST, containing the statistics. The first six locations of STATS contain the sum, median, lower hinge, upper hinge, minimum, and maximum values, respectively. When NST is set to 11, the extra 5 locations contain the mean, standard deviation, sample variance, skewness, and kurtosis, respectively.

15.3 EXAMPLES

The example executions of the SMFDST routine are contained on page C-9 and consist of an analysis of the Time-Between-Failures (TBF) data set and the fault counts for interval data (with equal lengths) data set introduced in Section 1.4. Table 15-1 shows the positions on the appendix page to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

TABLE 15-1. DATA STATISTICS EXAMPLE PAGES

ANALYSIS OF THE TBF DATA SET	C-9
ANALYSIS OF THE ERROR COUNTS OF THE INTERVAL DATA SET	C-9

CHAPTER 16

UTILITY ROUTINE FOR MODEL FIT ANALYSIS

16.1 INTRODUCTION

This chapter describes the SMFLIB utility routine, SMFGOF, which is used to calculate the chi-square Goodness-of-Fit statistic for interval data analysis.

Starting with the first interval, adjacent intervals are combined with the first until the expected frequency count is greater than or equal to the user-specified cell combination (CCN). Once this condition is satisfied, the procedure moves on to the next interval and repeats the process. This is continued until either all newly constructed intervals have the desired expected frequencies or all but the last interval achieve the desired counts.

The usual chi-square statistic is then calculated as:

$$CHISQU = \sum_{I=1}^M \frac{(DAT_I - PDAT_I)^2}{PDAT_I}$$

where $DAT(I)$ is the observed number of faults falling in the I^{th} interval, $PDAT(I)$ is the predicted (expected) fault count for the I^{th} interval based upon the chosen model, and M is the resulting number of intervals.

The degrees-of-freedom associated with the chi-square statistic is calculated as:

$$DOF = M - 1 - EST$$

where M is as previously defined, and EST is the number of parameters estimated in the model.

16.2 ACCESS LINE AND ARGUMENT LIST

16.2.1 SMFGOF Routine

The access line to the SMFGOF routine is as follows:

```
CALL SMFGOF (DAT      ,PDAT      ,NS      ,EST      ,CCN      ,CHI
              ,DOF      ,LIF      ,RFLAG   )
```

where the arguments of the call line, in the order of occurrence, are defined as:

DAT = Input real vector, of length greater than or equal to **NS**, containing the observed interval fault counts.

- PDAT** = Input real vector, of length greater than or equal to NS, containing the predicted interval fault counts from the model.
- NS** = Input integer containing the number of testing intervals conducted to date.
- EST** = Input integer containing the number of parameters estimated in the chosen software reliability model.
- CCN** = Input real containing the cell combination frequency number. The usual value of CCN is five; however, the user may execute the chi-square with no cells combined by setting CCN to minus one.
- CHI** = Output real containing the chi-square statistic. This value should only be considered final when RFLAG has a value of zero or one.
- DOF** = Output real containing the degrees-of-freedom. This value should only be considered final when RFLAG has a value of zero or one.
- LIF** = Output real containing the frequency of the last interval if the routine could not construct a partition for the last interval which satisfied the CCN value. This value should only be used when RFLAG has a value of one.
- RFLAG** = Output integer flag indicating the reason for the return to the calling program, where zero indicates successful processing. One (also) indicates successful processing; however, the CCN could not be achieved for the last interval (i.e., the argument LIF should be examined). Two indicates that the degrees-of-freedom was calculated to be less than one.

16.3 EXAMPLES

The example executions of the SMFGOF routine are contained on pages C-10 through C-12 and consist of an analysis of each of the interval models. All the Goodness-of-Fit analyses treat the cells individually. Table 16-1 shows the correlation of the appendix pages to the example executions. These examples were obtained from an actual SMERFS execution on the VAX 11/785 in which a "patch" was added to list the arguments before and after the SMFLIB routine processing. Arguments that retain their initially assigned values are not listed in the lower half of the examples.

TABLE 16-1. MODEL FIT ANALYSIS EXAMPLE PAGES

BROOKS AND MOTLEY MODEL FITS	C-10
GENERALIZED POISSON MODEL FITS	C-10
NON-HOMOGENEOUS POISSON MODEL FITS	C-11
SCHNEIDEWIND MAXIMUM LIKELIHOOD MODEL FIT	C-11
S-SHAPED RELIABILITY GROWTH MODEL FIT	C-12

CHAPTER 17

REFERENCES

1. Farr, William H., A Survey of Software Reliability Modeling and Estimation, NSWC TR 82-171, Sep 1983, NSWCDD, Dahlgren, VA.
2. Farr, William H. and Smith, Oliver D., Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) User's Guide, NSWCDD TR 84-373, Rev. 3, Sep 1993, NSWCDD, Dahlgren, VA.
3. Kraus, Paul and Vandergraft, James, Starting Values for the SMERFS Software Models, CE 25-90-23/jsn, 1107-03, Mar 1990.
4. Musa, John; Ianmino, A.; and Okumoto, K., Software Reliability Measurement. Prediction. Application, McGraw Hill, 1987.
5. Yamada, S.; Ohba, M.; and Osaki, S., "S-Shaped Reliability Growth Modeling for Software Error Detection," IEEE Transactions on Reliability, Vol. R-32, Dec 1983.
6. Littlewood, Bev; Abdel Ghaly, A. A.; and Chan, P. Y., "Tools for the Analysis of the Accuracy of Software Reliability Predictions," Software System Design Methods, Edited by J. K. Skwirzynski, NATO ASI Series, Vol. F22, Springer-Verlag, 1986, pp. 299-333.
7. Recommended Practice. Software Reliability, ANSI/AIAA R-013-1992, 23 Feb 1993.

APPENDIX A
EXAMPLES
FOR EXECUTION TIME DATA
ANALYSES

If the SGEOMA routine is accessed with:

DAT = (as defined in Table 1-1)
 NS = 29
 NSB = 14
 NSE = 28
 NSR = 15
 TYP = 1 (accuracy)

then the call to SGEOMA yields:

STAT = 0.6665782526E+02
 RFLAG = 0
 INDX = 15
 V = (
 0.3709955579E+01, 0.3890953301E+01, 0.3633806136E+01,
 0.4113845440E+01, 0.4221487743E+01, 0.4111638138E+01,
 0.4408220640E+01, 0.4378010962E+01, 0.4608033484E+01,
 0.4745807691E+01, 0.4970508422E+01, 0.4902294712E+01,
 0.4973945907E+01, 0.4957605075E+01, 0.5031712032E+01)
 VPRE = (not applicable for analysis type 1)

If the SGEOMA routine is accessed with the following change:

TYP = 2 (bias)

then the call to SGEOMA yields:

STAT = 0.4223024410E+00
 RFLAG = 0
 INDX = 15
 V = (
 0.4223024410E+00, 0.4604391803E+00, 0.4630772226E+00,
 0.4918271227E+00, 0.5166718840E+00, 0.5215157895E+00,
 0.5374079919E+00, 0.5771027970E+00, 0.5842988697E+00,
 0.5877466486E+00, 0.6052310332E+00, 0.6091894416E+00,
 0.6195051130E+00, 0.6226280885E+00, 0.6244392466E+00)
 VPRE = (
 0.6091894416E+00, 0.6195051130E+00, 0.4223024410E+00,
 0.6226280885E+00, 0.6052310332E+00, 0.4918271227E+00,
 0.5877466486E+00, 0.5166718840E+00, 0.5771027970E+00,
 0.5842988697E+00, 0.6244392466E+00, 0.5374079919E+00,
 0.5215157895E+00, 0.4630772226E+00, 0.4604391803E+00)

If the SGEOMA routine is accessed with the following change:

TYP = 3 (noise)

then the call to SGEOMA yields:

STAT = 0.1752071137E+01
 RFLAG = 0
 INDX = 15
 V = (not applicable for analysis type 3)
 VPRE = (not applicable for analysis type 3)

If the SGEOMA routine is accessed with the following change:

TYP = 4 (trend)

then the call to SGEOMA yields:

STAT = 0.1065625429E+00
 RFLAG = 0
 INDX = 15
 V = (
 0.7756203941E-01, 0.1573324114E+00, 0.2026301213E+00,
 0.2830808640E+00, 0.3598109450E+00, 0.4156944379E+00,
 0.4888468492E+00, 0.5488684335E+00, 0.6199164605E+00,
 0.6923813239E+00, 0.7732292295E+00, 0.8368708244E+00,
 0.8977239359E+00, 0.9490642758E+00, 0.1000000000E+01)
 VPRE = (not applicable for analysis type 4)

If the SGEOMD routine is accessed with:

ESF = 1 (Maximum Likelihood)
 NS = 29
 DAT = (as defined in Table 1-1)

then the call to SGEOMD yields:

STATS = (0.8756428653E+00,	0.7443611541E+00,	0.1000000000E+01)
	(0.5272965857E+00,	0.1472457174E-01,	0.1039868600E+01)
	(0.8921988353E+02,	0.2289763338E+02,	0.1555421337E+03)
	(0.9787439096E+00,	0.9520650059E+00,	0.1000000000E+01)
RFLAG =	0		

If the SGEOPR routine is accessed with:

D = 0.5272965857E+00
 PCON = 0.8756428653E+00
 NS = 29
 DAT = (as defined in Table 1-1)

then the call to SGEOPR yields:

PDAT = (0.1896465912E+01,	0.2165798394E+01,	0.2473380964E+01,
	0.2824645825E+01,	0.3225796654E+01,	0.3683918161E+01,
	0.4207101214E+01,	0.4804585728E+01,	0.5486923858E+01,
	0.6266166351E+01,	0.7156075381E+01,	0.8172367599E+01,
	0.9332991706E+01,	0.1065844544E+02,	0.1217213760E+02,
	0.1390080143E+02,	0.1587496682E+02,	0.1812949942E+02,
	0.2070421646E+02,	0.2364458991E+02,	0.2700254961E+02,
	0.3083740036E+02,	0.3521686932E+02,	0.4021830214E+02,
	0.4593002893E+02,	0.5245292431E+02,	0.5990218888E+02,
	0.6840938384E+02,	0.7812475446E+02)	
DVAL =	0.4104494580E+00		
DFLG =	1		
V = (0.3698034305E+00,	0.3718657819E+00,	0.4794149750E+00,
	0.4926860730E+00,	0.5006006418E+00,	0.5304155347E+00,
	0.5731778743E+00,	0.5885848309E+00,	0.5966652196E+00,
	0.5995259352E+00,	0.6135579733E+00,	0.6240089006E+00,
	0.6360585246E+00,	0.6437225289E+00,	0.6452830441E+00,
	0.6480621471E+00,	0.6516658892E+00,	0.6542640886E+00,
	0.6707724740E+00,	0.6846049082E+00,	0.6872821173E+00,
	0.6941420888E+00,	0.7010513657E+00,	0.7083855073E+00,
	0.7109256108E+00,	0.7131531816E+00,	0.7210430118E+00,
	0.7260718325E+00,	0.7646029266E+00)	

If the SGEOMD routine is accessed with:

ESF = 2 (Least Squares)
 NS = 29
 DAT = (as defined in Table 1-1)

then the call to SGEOMD yields:

STATS = (0.9048864049E+00,	0.0000000000E+00,	0.0000000000E+00)
	(0.2741500134E+00,	0.0000000000E+00,	0.0000000000E+00)
	(0.6618870597E+02,	0.0000000000E+00,	0.0000000000E+00)
	(0.9448903265E+00,	0.0000000000E+00,	0.0000000000E+00)
RFLAG =	0		

If the SGEOPR routine is accessed with:

D = 0.2741500134E+00
 PCON = 0.9048864049E+00
 NS = 29
 DAT = (as defined in Table 1-1)

then the call to SGEOPR yields:

PDAT = (0.3647637976E+01,	0.4031045174E+01,	0.4454752721E+01,
	0.4922996629E+01,	0.5440458163E+01,	0.6012310644E+01,
	0.6644271160E+01,	0.7342657735E+01,	0.8114452483E+01,
	0.8967371417E+01,	0.9909941589E+01,	0.1095158634E+02,
	0.1210271950E+02,	0.1337484952E+02,	0.1478069451E+02,
	0.1633430941E+02,	0.1805122645E+02,	0.1994861051E+02,
	0.2204543068E+02,	0.2436264990E+02,	0.2692343455E+02,
	0.2975338606E+02,	0.3288079686E+02,	0.3633693322E+02,
	0.4015634783E+02,	0.4437722526E+02,	0.4904176372E+02,
	0.5419659689E+02,	0.5989326019E+02)	
DVAL =	0.3186202771E+00		
DPLG =	1		
V = (0.2196982851E+00,	0.2409685553E+00,	0.3401966952E+00,
	0.4220685563E+00,	0.4294750504E+00,	0.4522971532E+00,
	0.4563148342E+00,	0.4856106320E+00,	0.5065636368E+00,
	0.5511380388E+00,	0.5583081506E+00,	0.5606413195E+00,
	0.5732320441E+00,	0.5776729222E+00,	0.5799805418E+00,
	0.5837373275E+00,	0.5902152227E+00,	0.6097896634E+00,
	0.6375385997E+00,	0.6405205657E+00,	0.6609595646E+00,
	0.6677856756E+00,	0.6704642458E+00,	0.6718455552E+00,
	0.6722328330E+00,	0.6762870476E+00,	0.6782633246E+00,
	0.6852097648E+00,	0.7328222583E+00)	

If the SJANMA routine is accessed with:

DAT = (as defined in Table 1-1)
 NS = 29
 NSB = 14
 NSE = 28
 NSR = 15
 TYP = 1 (accuracy)

then the call to SJANMA yields:

STAT = 0.7623093602E+02
 RFLAG = 0
 INDX = 15
 V = (
 0.4006590803E+01, 0.4227010139E+01, 0.4332138170E+01,
 0.4349038924E+01, 0.4593528871E+01, 0.4752855313E+01,
 0.4823277320E+01, 0.4990639201E+01, 0.5101993818E+01,
 0.5319704287E+01, 0.5589336049E+01, 0.5957219898E+01,
 0.6058511524E+01, 0.6107243965E+01, 0.6021847737E+01)
 VPRE = (not applicable for analysis type 1)

If the SJANMA routine is accessed with the following change:

TYP = 2 (bias)

then the call to SJANMA yields:

STAT = 0.6307424010E+00
 RFLAG = 0
 INDX = 15
 V = (
 0.1123262996E+00, 0.1277654702E+00, 0.1329409412E+00,
 0.1335198415E+00, 0.1737212635E+00, 0.2034241427E+00,
 0.2155420711E+00, 0.2257338644E+00, 0.2358593401E+00,
 0.2589568832E+00, 0.2870340453E+00, 0.3050882708E+00,
 0.3207769870E+00, 0.3377645334E+00, 0.3692575990E+00)
 VPRE = (
 0.3377645334E+00, 0.3207769870E+00, 0.1737212635E+00,
 0.3692575990E+00, 0.3050882708E+00, 0.2034241427E+00,
 0.2870340453E+00, 0.2155420711E+00, 0.2589568832E+00,
 0.2358593401E+00, 0.2257338644E+00, 0.1335198415E+00,
 0.1277654702E+00, 0.1123262996E+00, 0.1329409412E+00)

If the SJANMA routine is accessed with the following change:

TYP = 3 (noise)

then the call to SJANMA yields:

STAT = 0.3512244821E+01
 RFLAG = 0
 INDX = 15
 V = (not applicable for analysis type 3)
 VPRE = (not applicable for analysis type 3)

If the SJANMA routine is accessed with the following change:

TYP = 4 (trend)

then the call to SJANMA yields:

STAT = 0.2000601554E+00
 RFLAG = 0
 INDX = 15
 V = (
 0.1033057412E+00, 0.2002626693E+00, 0.2480944864E+00,
 0.3636133205E+00, 0.4548463376E+00, 0.5118547874E+00,
 0.5966586532E+00, 0.6575095832E+00, 0.7326316516E+00,
 0.8000601792E+00, 0.8641890678E+00, 0.9001127461E+00,
 0.9343772690E+00, 0.9642437335E+00, 0.1000000000E+01)
 VPRE = (not applicable for analysis type 4)

If the SJAMSD routine is accessed with:

ESF = 1 (Maximum Likelihood)
 NS = 29
 DAT = (as defined in Table 1-1)

then the call to SJAMSD yields:

STATS = (0.5800316041E-02,	0.1711692571E+00,	0.2960091936E-02,
	0.2951033287E+02,	0.5103328706E+00,	0.3378273451E+03)
RFLAG =	0		

If the SJAMPR routine is accessed with:

NPV = 29 (vector generation)
 NS = 29
 PCON = 0.5800316041E-02
 TNOF = 0.2951033287E+02
 EXE = 0
 DAT = (as defined in Table 1-1)

then the call to SJAMPR yields:

PDAT = (0.5842170591E+01,	0.6047084739E+01,	0.6266896138E+01,
	0.6503290610E+01,	0.6758218315E+01,	0.7033947671E+01,
	0.7333133043E+01,	0.7658900462E+01,	0.8014957270E+01,
	0.8405733827E+01,	0.8836568805E+01,	0.9313954537E+01,
	0.9845866443E+01,	0.1044221217E+02,	0.1111545447E+02,
	0.1188149165E+02,	0.1276092902E+02,	0.1378096015E+02,
	0.1497822876E+02,	0.1640332432E+02,	0.1812811404E+02,
	0.2025824388E+02,	0.2295562684E+02,	0.2648165651E+02,
	0.3128747443E+02,	0.3822431820E+02,	0.4911340467E+02,
	0.6867790357E+02,	0.1141499349E+03)	
DVAL =	0.2328061439E+00		
DFLG =	2		
V = (0.1524190610E+00,	0.1990461823E+00,	0.2898922176E+00,
	0.2991204217E+00,	0.3289566658E+00,	0.3603158927E+00,
	0.3695395963E+00,	0.4204303557E+00,	0.4992430327E+00,
	0.5431509592E+00,	0.5471356638E+00,	0.5555793420E+00,
	0.6095168286E+00,	0.6139279628E+00,	0.6393960147E+00,
	0.6459839554E+00,	0.6512558201E+00,	0.7151364806E+00,
	0.7220261680E+00,	0.7406229510E+00,	0.7461809767E+00,
	0.7489626051E+00,	0.7801833078E+00,	0.7952577696E+00,
	0.7960793037E+00,	0.8045962601E+00,	0.8088865578E+00,
	0.8115814027E+00,	0.8162113134E+00)	

If the SJAMMD routine is accessed with:

ESF = 2 (Least Squares)
 NS = 29
 DAT = (as defined in Table 1-1)

then the call to SJAMMD yields:

STATS = (0.3203936698E-02, 0.1056993933E+00, 0.1278522907E-01,
 0.3299047493E+02, 0.3990474928E+01, 0.7821525879E+02)
 RFLAG = 0

If the SJAMPR routine is accessed with:

NPV = 1 (MTBNF to discover next K failures)
 NS = 29
 PCON = 0.3203936698E-02
 TNOF = 0.3299047493E+02
 EXE = 2
 DAT = (as defined in Table 1-1)

then the call to SJAMPR yields:

PDAT = (0.1825853126E+03)
 DVAL = (only applicable during vector generation)
 DFLG = (only applicable during vector generation)
 V = (only applicable during vector generation)

If the SJAMPR routine is accessed with the following change:

NPV = 29 (vector generation)

then the call to SJAMPR yields:

PDAT = (0.9460792239E+01, 0.9756530026E+01, 0.1007135353E+02,
 0.1040717194E+02, 0.1076615785E+02, 0.1115079433E+02,
 0.1156393246E+02, 0.1200886209E+02, 0.1248939966E+02,
 0.1300999793E+02, 0.1357588437E+02, 0.1419323731E+02,
 0.1486941245E+02, 0.1561323732E+02, 0.1643539882E+02,
 0.1734895996E+02, 0.1837005914E+02, 0.1951887174E+02,
 0.2082095669E+02, 0.2230918041E+02, 0.2402652951E+02,
 0.2603033083E+02, 0.2839877541E+02, 0.3124136054E+02,
 0.3471630049E+02, 0.3906101101E+02, 0.4464875883E+02,
 0.5210205083E+02, 0.6254234991E+02)
 DVAL = 0.2262206874E+00
 DFLG = 2
 V = (0.9741774948E-01, 0.1300551584E+00, 0.1905487958E+00,
 0.2008441329E+00, 0.2198185825E+00, 0.2504353815E+00,
 0.2924189463E+00, 0.3932455004E+00, 0.4028698510E+00,
 0.4057417847E+00, 0.4593113663E+00, 0.4796412876E+00,
 0.4879499227E+00, 0.5056593273E+00, 0.5575586891E+00,
 0.5966312265E+00, 0.5981483816E+00, 0.5985467510E+00,
 0.6098860945E+00, 0.6456698492E+00, 0.6589302354E+00,
 0.6808994421E+00, 0.6842266478E+00, 0.6990205589E+00,
 0.7073701682E+00, 0.7130994906E+00, 0.7234231857E+00,
 0.7392965355E+00, 0.7827399738E+00)

If the SLAVMA routine is accessed with:

DAT = (as defined in Table 1-1)
 NS = 29
 NSB = 14
 NSE = 28
 NSR = 15
 TYP = 1 (accuracy)
 MAXIC = 100
 PHIIND = 1 (Linear function)

then the call to SLAVMA yields:

STAT = 0.6867318398E+02
 RFLAG = 0
 INDX = 15
 V = (
 0.3811703904E+01, 0.4036179568E+01, 0.3485165630E+01,
 0.4299519219E+01, 0.4404488662E+01, 0.4087484460E+01,
 0.4597987220E+01, 0.4428287684E+01, 0.4820947030E+01,
 0.5009861254E+01, 0.5400531922E+01, 0.5100737925E+01,
 0.5149829989E+01, 0.4980950000E+01, 0.5059509508E+01)
 VPRE = (not applicable for analysis type 1)

If the SLAVMA routine is accessed with the following change:

TYP = 2 (bias)

then the call to SLAVMA yields:

STAT = 0.6668247965E+00
 RFLAG = 0
 INDX = 15
 V = (
 0.6239958470E+00, 0.7334914666E+00, 0.7778836296E+00,
 0.7827704998E+00, 0.7926798234E+00, 0.7968359945E+00,
 0.8062558481E+00, 0.8242595238E+00, 0.8244206911E+00,
 0.8287775339E+00, 0.8368205713E+00, 0.8388477557E+00,
 0.8409289687E+00, 0.8547550982E+00, 0.8929001517E+00)
 VPRE = (
 0.7827704998E+00, 0.8062558481E+00, 0.6239958470E+00,
 0.8244206911E+00, 0.8242595238E+00, 0.7334914666E+00,
 0.8287775339E+00, 0.7778836296E+00, 0.8388477557E+00,
 0.8547550982E+00, 0.8929001517E+00, 0.8409289687E+00,
 0.8368205713E+00, 0.7926798234E+00, 0.7968359945E+00)

If the SLAVMA routine is accessed with the following change:

TYP = 3 (noise)

then the call to SLAVMA yields:

STAT = 0.1234639489E+01
 RFLAG = 0
 INDX = 15
 V = (not applicable for analysis type 3)
 VPRE = (not applicable for analysis type 3)

If the SLAVMA routine is accessed with the following change:

TYP = 4 (trend)

then the call to SLAVMA yields:

STAT = 0.7342993142E-01
 RFLAG = 0
 INDX = 15
 V = (
 0.6101454965E-01, 0.1266014281E+00, 0.1656908032E+00,
 0.2352118806E+00, 0.3046962926E+00, 0.3575404656E+00,
 0.4280656833E+00, 0.4881911891E+00, 0.5611386906E+00,
 0.6382394087E+00, 0.7275150454E+00, 0.8009820052E+00,
 0.8734299433E+00, 0.9363103372E+00, 0.1000000000E+01)
 VPRE = (not applicable for analysis type 4)

If the SLAVMD routine is accessed with:

DAT = (as defined in Table 1-1)
 ESF = 1 (Maximum Likelihood)
 MAXIC = 100
 NS = 29
 N = 3
 PHIIND = 1 (Linear function)
 BETA = (0.3914285714E+01, 0.1957142857E+01)

then the call to SLAVMD yields:

ALPHA = 0.1125571613E+06
 COUNT = 101
 RFLAG = 1 (maximum iterations reached)
 X = (0.1000000000E-09, 0.1268820081E+06)
 Z = 0.1037313468E+03

If the SLAVMA routine is accessed with:

DAT = (as defined in Table 1-1)
 NS = 29
 NSB = 14
 NSE = 28
 NSR = 15
 TYP = 1 (accuracy)
 MAXIC = 100
 PHIIND = 2 (Quadratic function)

then the call to SLAVMA yields:

STAT = 0.6633489943E+02
 RFLAG = 0
 INDX = 15
 V = (
 0.3710657296E+01, 0.3898849528E+01, 0.3556969904E+01,
 0.4126459136E+01, 0.4229780054E+01, 0.4054022578E+01,
 0.4410826289E+01, 0.4332764278E+01, 0.4609088590E+01,
 0.4758402696E+01, 0.5033817961E+01, 0.4880980976E+01,
 0.4938769843E+01, 0.4861277041E+01, 0.4932233257E+01)
 VPRE = (not applicable for analysis type 1)

If the SLAVMA routine is accessed with the following change:

TYP = 2 (bias)

then the call to SLAVMA yields:

STAT = 0.5137304450E+00
 RFLAG = 0
 INDX = 15
 V = (
 0.4866584053E+00, 0.5803971152E+00, 0.6197983255E+00,
 0.6199670982E+00, 0.6216621599E+00, 0.6583072011E+00,
 0.6745163337E+00, 0.6788526757E+00, 0.6818462544E+00,
 0.6822147207E+00, 0.6849332018E+00, 0.6869451621E+00,
 0.6895087688E+00, 0.7040747281E+00, 0.7533949133E+00)
 VPRE = (
 0.6583072011E+00, 0.6788526757E+00, 0.4866584053E+00,
 0.6895087688E+00, 0.6849332018E+00, 0.5803971152E+00,
 0.6818462544E+00, 0.6197983255E+00, 0.6869451621E+00,
 0.7040747281E+00, 0.7533949133E+00, 0.6822147207E+00,
 0.6745163337E+00, 0.6199670982E+00, 0.6216621599E+00)

If the SLAVMA routine is accessed with the following change:

TYP = 3 (noise)

then the call to SLAVMA yields:

STAT = 0.1390964398E+01
 RFLAG = 0
 INDX = 15
 V = (not applicable for analysis type 3)
 VPRE = (not applicable for analysis type 3)

If the SLAVMA routine is accessed with the following change:

TYP = 4 (trend)

then the call to SLAVMA yields:

STAT = 0.8005099686E-01
 RFLAG = 0
 INDX = 15
 V = (
 0.6641336475E-01, 0.1366619571E+00, 0.1779020068E+00,
 0.2502375643E+00, 0.3216683695E+00, 0.3753786843E+00,
 0.4462064826E+00, 0.5060152829E+00, 0.5778422948E+00,
 0.6531494925E+00, 0.7397324532E+00, 0.8106319198E+00,
 0.8800510088E+00, 0.9398872690E+00, 0.1000000000E+01)
 VPRE = (not applicable for analysis type 4)

If the SLAVMD routine is accessed with:

```
DAT = (as defined in Table 1-1)
ESP = 1 (Maximum Likelihood)
MAXIC = 100
NS = 29
N = 3
PHIIND = 2 (Quadratic function)
BETA = ( 0.5727057106E+00, 0.6567989882E-01)
```

then the call to SLAVMD yields:

```
ALPHA = 0.8782648974E+05
COUNT = 30
RFLAG = 0
X = ( 0.1238759877E+06, 0.5324349831E+04)
Z = 0.1011822240E+03
```

If the SLAVPR routine is accessed with:

```
NPV = 1 (MTBNF to discover next failure)
NS = 29
ALPHA = 0.8782648974E+05
BETA0 = 0.1238759877E+06
BETA1 = 0.5324349831E+04
PHI = 2
DAT = (as defined in Table 1-1)
```

then the call to SLAVPR yields:

```
PDAT = ( 0.5597225646E+02)
DVAL = (only applicable during vector generation)
DFLG = (only applicable during vector generation)
V = (only applicable during vector generation)
```

If the SLAVPR routine is accessed with the following change:

```
NPV = 29 (vector generation)
```

then the call to SLAVPR yields:

```
PDAT = ( 0.1471102955E+01, 0.1652975547E+01, 0.1956096535E+01,
          0.2380465917E+01, 0.2926083694E+01, 0.3592949866E+01,
          0.4381064433E+01, 0.5290427394E+01, 0.6321038751E+01,
          0.7472898502E+01, 0.8746006649E+01, 0.1014036319E+02,
          0.1165596813E+02, 0.1329282146E+02, 0.1505092318E+02,
          0.1693027330E+02, 0.1893087182E+02, 0.2105271873E+02,
          0.2329581404E+02, 0.2566015774E+02, 0.2814574983E+02,
          0.3075259032E+02, 0.3348067921E+02, 0.3633001649E+02,
          0.3930060216E+02, 0.4239243623E+02, 0.4560551870E+02,
          0.4893984956E+02, 0.5239542881E+02)
DVAL = 0.4194288524E+00
DFLG = 1
V = ( 0.4010846006E+00, 0.4539116109E+00, 0.4694760405E+00,
       0.5013307651E+00, 0.5508375889E+00, 0.5588608245E+00,
       0.5628678975E+00, 0.5976762052E+00, 0.5986918833E+00,
       0.6081419977E+00, 0.6211386180E+00, 0.6221934719E+00,
       0.6308758717E+00, 0.6423919370E+00, 0.6546475968E+00,
       0.6555774246E+00, 0.6565628804E+00, 0.6571769127E+00,
       0.6580736218E+00, 0.6638462508E+00, 0.6731622232E+00,
       0.6777034044E+00, 0.6782979468E+00, 0.6852807772E+00,
       0.6859523341E+00, 0.7164205071E+00, 0.7214230027E+00,
       0.7403926150E+00, 0.7432183358E+00)
```

If the SLAVMD routine is accessed with:

DAT = (as defined in Table 1-1)
ESF = 2 (Least Squares)
MAXIC = 0
NS = 29
N = 3
PHIIND = 1 (Linear function)
BETA = (0.0000000000E+00, 0.0000000000E+00)

then the call to SLAVMD yields:

ALPHA = 0.2000000000E+01
COUNT = 0
RFLAG = 4 (estimates deemed invalid)
X = (-0.9408866995E+01, 0.1957142857E+01)
Z = 0.9304438424E+03

If the SLAVMD routine is accessed with:

DAT = (as defined in Table 1-1)
 ESP = 2 (Least Squares)
 MAXIC = 0
 NS = 29
 N = 3
 PHIIND = 2 (Quadratic function)
 BETA = (0.0000000000E+00, 0.0000000000E+00)

then the call to SLAVMD yields:

ALPHA = 0.2000000000E+01
 COUNT = 0
 RFLAG = 0
 X = (0.5727057106E+00, 0.6567989882E-01)
 Z = 0.3361231481E+03

If the SLAVPR routine is accessed with:

NPV = 1 (MTBNF to discover next failure)
 NS = 29
 ALPHA = 0.2000000000E+01
 BETAO = 0.5727057106E+00
 BETAL = 0.6567989882E-01
 PHI = 2
 DAT = (as defined in Table 1-1)

then the call to SLAVPR yields:

PDAT = (0.5968461465E+02)
 DVAL = (only applicable during vector generation)
 DFLG = (only applicable during vector generation)
 V = (only applicable during vector generation)

If the SLAVPR routine is accessed with the following change:

NPV = 29 (vector generation)

then the call to SLAVPR yields:

PDAT = (0.6383856095E+00, 0.8354253059E+00, 0.1163824800E+01,
 0.1623584092E+01, 0.2214703181E+01, 0.2937182068E+01,
 0.3791020753E+01, 0.4776219235E+01, 0.5892777515E+01,
 0.7140695592E+01, 0.8519973468E+01, 0.1003061114E+02,
 0.1167260861E+02, 0.1344596588E+02, 0.1535068294E+02,
 0.1738675981E+02, 0.1955419647E+02, 0.2185299293E+02,
 0.2428314918E+02, 0.2684466524E+02, 0.2953754109E+02,
 0.3236177674E+02, 0.3531737219E+02, 0.3840432743E+02,
 0.4162264247E+02, 0.4497231731E+02, 0.4845335195E+02,
 0.5206574638E+02, 0.5580950062E+02)
 DVA = 0.6162241748E+00
 DFLG = 1
 V = (0.6159695173E+00, 0.6445464766E+00, 0.6851896917E+00,
 0.6974691921E+00, 0.6986338044E+00, 0.7081807181E+00,
 0.7125643144E+00, 0.7265459942E+00, 0.7269787166E+00,
 0.7362313830E+00, 0.7441894404E+00, 0.7538686753E+00,
 0.7572198848E+00, 0.7572406377E+00, 0.7581657858E+00,
 0.7585897462E+00, 0.7626393062E+00, 0.7632316752E+00,
 0.7660175840E+00, 0.7718599772E+00, 0.7738988440E+00,
 0.7775720377E+00, 0.7849797311E+00, 0.7928228654E+00,
 0.8035574016E+00, 0.8065052648E+00, 0.8766917730E+00,
 0.8990965957E+00, 0.9414550232E+00)

If the SMUSMA routine is accessed with:

DAT = (as defined in Table 1-1)
 NS = 29
 NSB = 14
 NSE = 28
 NSR = 15
 TYP = 1 (accuracy)

then the call to SMUSMA yields:

STAT = 0.7141991922E+02
 RFLAG = 0
 INDX = 15
 V = (0.3910987296E+01, 0.4118393179E+01, 0.3863305847E+01,
 0.4325188539E+01, 0.4472663849E+01, 0.4392908888E+01,
 0.4675590793E+01, 0.4678290821E+01, 0.4906330453E+01,
 0.5077258406E+01, 0.5334353534E+01, 0.5351254230E+01,
 0.5431685539E+01, 0.5420753027E+01, 0.5460954817E+01)
 VPRE = (not applicable for analysis type 1)

If the SMUSMA routine is accessed with the following change:

TYP = 2 (bias)

then the call to SMUSMA yields:

STAT = 0.4466545502E+00
 RFLAG = 0
 INDX = 15
 V = (0.2934109911E+00, 0.3033036044E+00, 0.3303836919E+00,
 0.3432722544E+00, 0.3562308921E+00, 0.3901699591E+00,
 0.3990966021E+00, 0.4298436630E+00, 0.4356180959E+00,
 0.4451705031E+00, 0.4750255816E+00, 0.5024942853E+00,
 0.5371339003E+00, 0.5508845178E+00, 0.5533454498E+00)
 VPRE = (0.5508845178E+00, 0.5371339003E+00, 0.3562308921E+00,
 0.5533454498E+00, 0.5024942853E+00, 0.3901699591E+00,
 0.4750255816E+00, 0.3990966021E+00, 0.4451705031E+00,
 0.4298436630E+00, 0.4356180959E+00, 0.3432722544E+00,
 0.3303836919E+00, 0.2934109911E+00, 0.3033036044E+00)

If the SMUSMA routine is accessed with the following change:

TYP = 3 (noise)

then the call to SMUSMA yields:

STAT = 0.2655320452E+01
 RFLAG = 0
 INDX = 15
 V = (not applicable for analysis type 3)
 VPRE = (not applicable for analysis type 3)

If the SMUSMA routine is accessed with the following change:

TYP = 4 (trend)

then the call to SMUSMA yields:

STAT = 0.1529756136E+00
 RFLAG = 0
 INDX = 15
 V = (0.9510402335E-01, 0.1866250201E+00, 0.2389505024E+00,
 0.3347073333E+00, 0.4176539430E+00, 0.4764141258E+00,
 0.5529756196E+00, 0.6134877798E+00, 0.6834777660E+00,
 0.7502302249E+00, 0.8181920976E+00, 0.8681497778E+00,
 0.9157983569E+00, 0.9570616044E+00, 0.1000000000E+01)
 VPRE = (not applicable for analysis type 4)

If the SMUSMD routine is accessed with:

NS = 29
 DAT = (as defined in Table 1-1)
 XLTH = 0.3000000000E+02

then the call to SMUSMD yields:

CMTBF = 0.1377044932E+03
 ECR = 0.1498021624E-01
 IMTBF = (0.6182099307E+01, 0.3027215670E+01, 0.1935483326E+02)
 IFLG = 1
 TNOF = (0.3015243123E+02, 0.2900000000E+02, 0.4107080476E+02)
 IIF = 0.1617573498E+00
 CIF = 0.7261927167E-02
 RFLAG = 0

If the SMUSPR routine is accessed with:

NPV = 2 (future reliability & additional testing time predictions)
 NS = 29
 DAT = (as defined in Table 1-1)
 IMTBF = 0.6182099307E+01
 CMTBF = 0.1377044932E+03
 DMTBF = 0.3600000000E+04 (one hour in seconds)
 TNOF = 0.3015243123E+02
 IIF = 0.1617573498E+00

then the call to SMUSPR yields:

PDAT = (0.1152431226E+01, 0.6083485181E+03)
 DVAL = (only applicable during vector generation)
 DFLG = (only applicable during vector generation)
 V = (only applicable during vector generation)

If the SMUSPR routine is accessed with the following change:
 NPV = 29 (vector generation)

then the call to SMUSPR yields:

PDAT = (0.6182099307E+01, 0.6248786061E+01, 0.6282398714E+01,
 0.6367223495E+01, 0.6470526387E+01, 0.6522804645E+01,
 0.6610875381E+01, 0.6754268615E+01, 0.6975211252E+01,
 0.7222729026E+01, 0.7539456460E+01, 0.7827965697E+01,
 0.8237252690E+01, 0.8855951342E+01, 0.9394279038E+01,
 0.1018148318E+02, 0.1121367999E+02, 0.1195931299E+02,
 0.1349359342E+02, 0.1543027290E+02, 0.1727031408E+02,
 0.2028595323E+02, 0.2357387222E+02, 0.2867279739E+02,
 0.3591892444E+02, 0.4773147065E+02, 0.6175002108E+02,
 0.8118184613E+02, 0.1047433110E+03)
 DVAL = 0.2379695003E+00
 DFLG = 2
 V = (0.1475174965E+00, 0.2061733507E+00, 0.2751462334E+00,
 0.3166253427E+00, 0.3265089496E+00, 0.3468689770E+00,
 0.3733654728E+00, 0.4033985701E+00, 0.4504293082E+00,
 0.5144938153E+00, 0.5827970927E+00, 0.5879387169E+00,
 0.5979714815E+00, 0.5998081965E+00, 0.6452719735E+00,
 0.6618220724E+00, 0.6937003052E+00, 0.7006566367E+00,
 0.7224076761E+00, 0.7231615970E+00, 0.7239295471E+00,
 0.7305864636E+00, 0.7549183726E+00, 0.7843636474E+00,
 0.7942217863E+00, 0.7990116894E+00, 0.8145999712E+00,
 0.8233991584E+00, 0.8300527570E+00)

NSWCDD TR 84-371

If the SMUSCT routine is accessed with:

TNOF	=	0.3015243123E+02	
CMTBFH	=	0.3825124811E-01	(converted to hours)
DMTBFH	=	0.4000000000E+02	(specified in hours)
IMTBFH	=	0.1717249807E-02	(converted to hours)
NS	=	29	
PC	=	0.2000000000E+01	
PF	=	0.2000000000E+01	
PI	=	0.5000000000E+01	
PMQ	=	0.9500000000E+00	
RHOC	=	0.8000000000E+00	
TC	=	0.1240000000E+01	
TI	=	0.2400000000E+01	
XMC	=	0.1850000000E+01	
XMF	=	0.8780000000E+01	
XMI	=	0.7210000000E+01	
XMQ	=	0.3000000000E+01	

then the call to SMUSCT yields:

XINT	=	0.1726129701E+02
------	---	------------------

If the SMSAMA routine is accessed with:

DAT = (as defined in Table 1-1)
 NS = 29
 NSB = 14
 NSE = 28
 NSR = 15
 TYP = 1 (accuracy)

then the call to SMSAMA yields:

STAT = 0.6730059365E+02
 RFLAG = 0
 INDX = 15
 V = (0.3787855313E+01, 0.3974035815E+01, 0.3613227718E+01,
 0.4196145807E+01, 0.4292981614E+01, 0.4125955302E+01,
 0.4468661596E+01, 0.4403211999E+01, 0.4661581424E+01,
 0.4802397368E+01, 0.5045420143E+01, 0.4938128372E+01,
 0.5002117305E+01, 0.4959487203E+01, 0.5029386670E+01)
 VPRE = (not applicable for analysis type 1)

If the SMSAMA routine is accessed with the following change:

TYP = 2 (bias)

then the call to SMSAMA yields:

STAT = 0.4764891276E+00
 RFLAG = 0
 INDX = 15
 V = (0.4764891276E+00, 0.5073901059E+00, 0.5077138033E+00,
 0.5426415879E+00, 0.5638443338E+00, 0.5660685015E+00,
 0.5785303613E+00, 0.6247576843E+00, 0.6293678133E+00,
 0.6369369093E+00, 0.6547686164E+00, 0.6647315907E+00,
 0.6663674811E+00, 0.6719474204E+00, 0.6759625124E+00)
 VPRE = (0.6663674811E+00, 0.6719474204E+00, 0.4764891276E+00,
 0.6759625124E+00, 0.6547686164E+00, 0.5426415879E+00,
 0.6369369093E+00, 0.5660685015E+00, 0.6247576843E+00,
 0.6293678133E+00, 0.6647315907E+00, 0.5785303613E+00,
 0.5638443338E+00, 0.5073901059E+00, 0.5077138033E+00)

If the SMSAMA routine is accessed with the following change:

TYP = 3 (noise)

then the call to SMSAMA yields:

STAT = 0.1806981956E+01
 RFLAG = 0
 INDX = 15
 V = (not applicable for analysis type 3)
 VPRE = (not applicable for analysis type 3)

If the SMSAMA routine is accessed with the following change:

TYP = 4 (trend)

then the call to SMSAMA yields:

STAT = 0.1088510819E+00
 RFLAG = 0
 INDX = 15
 V = (0.7922129330E-01, 0.1596598111E+00, 0.2063675769E+00,
 0.2876948367E+00, 0.3644497652E+00, 0.4209069089E+00,
 0.4940272668E+00, 0.5542791188E+00, 0.6250182252E+00,
 0.6966494758E+00, 0.7755177684E+00, 0.8378725564E+00,
 0.8977554410E+00, 0.9488540012E+00, 0.1000000000E+01)
 VPRE = (not applicable for analysis type 4)

```

NS      =      29
DAT      =      (as defined in Table 1-1)
SUMTBF  =      0.6085000000E+03

```

then the call to SMSAND yields:

```

BETA0      =      0.7844651420E+01
BETA1      =      0.6461385050E-01
CIF        =      0.1320700508E-01
IIF        =      0.5068731340E+00
RFLAG      =      0

```

If the SMSAPR routine is accessed with:

```

NPV      =      1 (expected number of failures during additional testing)
NS       =      29
CIF      =      0.1320700508E-01
DIF      =      0.0000000000E+00
BETA0    =      0.7844651420E+01
BETA1    =      0.6461385050E-01
EXTM     =      0.4178500000E+04 (one hour in seconds plus the time of
                                the last failure)
DAT      =      (as defined in Table 1-1)

```

then the call to SMSAPR yields:

```

PDAT      = (      0.1954564768E+01)
DVAL      =      (only applicable during vector generation)
DFLG      =      (only applicable during vector generation)
V         =      (only applicable during vector generation)

```

If the SMSAPR routine is accessed with the following changes:

```
NPV      =      2 (failures and testing time to reach desired hazard rate)
DIF      =      0.2777777778E-03 (one failure per hour in seconds)
```

then the call to SMSAPR yields:

```

PDAT      = (      0.3029354309E+02,      0.2764676855E+05)
DVAL      =      (only applicable during vector generation)
DFLG      =      (only applicable during vector generation)
V         =      (only applicable during vector generation)

```

NSWCDD TR 84-371

If the SMSAPR routine is accessed with the following change:

NPV = 29 (vector generation)

then the call to SMSAPR yields:

PDAT = (0.2261117035E+01,	0.2591465054E+01,	0.2970076746E+01,
	0.3404003408E+01,	0.3901326529E+01,	0.4471308299E+01,
	0.5124564108E+01,	0.5873260249E+01,	0.6731340506E+01,
	0.7714785841E+01,	0.8841912025E+01,	0.1013371076E+02,
	0.1161424061E+02,	0.1331107510E+02,	0.1525581623E+02,
	0.1748468301E+02,	0.2003918608E+02,	0.2296690070E+02,
	0.2632235291E+02,	0.3016803493E+02,	0.3457556909E+02,
	0.3962704168E+02,	0.4541653179E+02,	0.5205186338E+02,
	0.5965661345E+02,	0.6837241354E+02,	0.7836158748E+02,
	0.8981017451E+02,	0.1029313941E+03)	
DVAL =	0.4365519406E+00		
DFLG =	1		
V = (0.3537021855E+00,	0.3787552648E+00,	0.5055174575E+00,
	0.5088821609E+00,	0.5123450933E+00,	0.5201701799E+00,
	0.5679695913E+00,	0.5734848672E+00,	0.5835928387E+00,
	0.5920007291E+00,	0.6145683726E+00,	0.6254850422E+00,
	0.6255411979E+00,	0.6304642254E+00,	0.6366393195E+00,
	0.6405395910E+00,	0.6494769576E+00,	0.6495730420E+00,
	0.6541735302E+00,	0.6755721791E+00,	0.6777632129E+00,
	0.6872826103E+00,	0.6888461344E+00,	0.6930506877E+00,
	0.7001081934E+00,	0.7104425333E+00,	0.7267966841E+00,
	0.7405038339E+00,	0.7539476707E+00)	

If the SNPTMA routine is accessed with:

DAT = (as defined in Table 1-1; however, converted to TTF data)
 NS = 29
 NSB = 14
 NSE = 28
 NSR = 15
 TYP = 1 (accuracy)

then the call to SNPTMA yields:

STAT = 0.7141991922E+02
 RFLAG = 0
 INDX = 15
 V = (
 0.3910987296E+01, 0.4118393179E+01, 0.3863305847E+01,
 0.4325188539E+01, 0.4472663849E+01, 0.4392908888E+01,
 0.4675590793E+01, 0.4678290821E+01, 0.4906330453E+01,
 0.5077258406E+01, 0.5334353534E+01, 0.5351254230E+01,
 0.5431685539E+01, 0.5420753027E+01, 0.5460954817E+01)
 VPRE = (not applicable for analysis type 1)

If the SNPTMA routine is accessed with the following change:

TYP = 2 (bias)

then the call to SNPTMA yields:

STAT = 0.4466545502E+00
 RFLAG = 0
 INDX = 15
 V = (
 0.2934109914E+00, 0.3033036046E+00, 0.3303836921E+00,
 0.3432722546E+00, 0.3562308921E+00, 0.3901699591E+00,
 0.3990966021E+00, 0.4298436629E+00, 0.4356180960E+00,
 0.4451705031E+00, 0.4750255816E+00, 0.5024942853E+00,
 0.5371339002E+00, 0.5508845178E+00, 0.5533454498E+00)
 VPRE = (
 0.5508845178E+00, 0.5371339002E+00, 0.3562308921E+00,
 0.5533454498E+00, 0.5024942853E+00, 0.3901699591E+00,
 0.4750255816E+00, 0.3990966021E+00, 0.4451705031E+00,
 0.4298436629E+00, 0.4356180960E+00, 0.3432722546E+00,
 0.3303836921E+00, 0.2934109914E+00, 0.3033036046E+00)

If the SNPTMA routine is accessed with the following change:

TYP = 3 (noise)

then the call to SNPTMA yields:

STAT = 0.2655320451E+01
 RFLAG = 0
 INDX = 15
 V = (not applicable for analysis type 3)
 VPRE = (not applicable for analysis type 3)

If the SNPTMA routine is accessed with the following change:

TYP = 4 (trend)

then the call to SNPTMA yields:

STAT = 0.1529756135E+00
 RFLAG = 0
 INDX = 15
 V = (
 0.9510402334E-01, 0.1866250200E+00, 0.2389505023E+00,
 0.3347073332E+00, 0.4176539429E+00, 0.4764141258E+00,
 0.5529756195E+00, 0.6134877797E+00, 0.6834777659E+00,
 0.7502302247E+00, 0.8181920975E+00, 0.8681497777E+00,
 0.9157983568E+00, 0.9570616044E+00, 0.1000000000E+01)
 VPRE = (not applicable for analysis type 4)

If the **SNPTMD** routine is accessed with:

DAT = (as defined in Table 1-1; however, converted to TTF data)
NS = 29

then the call to **SNPTMD** yields:

STATS = (0.5201767285E-02, 0.3050480265E+02)
RFLAG = 0

If the **SNPTPR** routine is accessed with:

NPV = 1 (program reliability for a specified operational time)
NS = 29
DAT = (as defined in Table 1-1; however, converted to TTF data)
PCON = 0.5201767285E-02
TNOF = 0.3050480265E+02
SOT = 0.3600000000E+04 (one hour in seconds)
SR = -0.1000000000E+01

then the call to **SNPTPR** yields:

PDAT = (0.2220611155E+00)
DVAL = (only applicable during vector generation)
DFLG = (only applicable during vector generation)
V = (only applicable during vector generation)

If the **SNPTPR** routine is accessed with the following change:

SR = 0.9500000000E+00 (additional time to reach a specified reliability for a specified operational time)

then the call to **SNPTPR** yields:

PDAT = (0.1228059432E+04)
DVAL = (only applicable during vector generation)
DFLG = (only applicable during vector generation)
V = (only applicable during vector generation)

If the **SNPTPR** routine is accessed with the following change:

NPV = 29 (vector generation)

then the call to **SNPTPR** yields:

PDAT = (0.6334874297E+01,	0.6367798122E+01,	0.6475173559E+01,
	0.6591692876E+01,	0.6567287694E+01,	0.6718023637E+01,
	0.6979622310E+01,	0.7389706809E+01,	0.7679673730E+01,
	0.8175541313E+01,	0.8278533171E+01,	0.9076884766E+01,
	0.1039568585E+02,	0.1030388661E+02,	0.1189781798E+02,
	0.1349644643E+02,	0.1230932950E+02,	0.1669034653E+02,
	0.1885635298E+02,	0.1848707268E+02,	0.2383997633E+02,
	0.2463819526E+02,	0.3101291958E+02,	0.3674080990E+02,
	0.4685111079E+02,	0.4747034587E+02,	0.5334923797E+02,
	0.5449366532E+02,	0.6103198814E+02)	
DVAL =	0.2321252429E+00		
DFLG =	2		
V = (0.1449786965E+00,	0.2029497005E+00,	0.2707310301E+00,
	0.3121272229E+00,	0.3216054230E+00,	0.3669291180E+00,
	0.3680970591E+00,	0.4230013888E+00,	0.4449034665E+00,
	0.5337976793E+00,	0.5769528353E+00,	0.5933504331E+00,
	0.5943057458E+00,	0.6049984634E+00,	0.6452104194E+00,
	0.6567433770E+00,	0.6896079772E+00,	0.6979776610E+00,
	0.7275612861E+00,	0.7291799861E+00,	0.7367105507E+00,
	0.7413677242E+00,	0.7632512746E+00,	0.7826064450E+00,
	0.7911508544E+00,	0.8036212613E+00,	0.8137794995E+00,
	0.8253775230E+00,	0.8308399954E+00)	

..

APPENDIX B
EXAMPLES
FOR INTERVAL DATA
ANALYSES

If the **SBAMGA** routine is accessed with:

```

CDAT  = (fault counts as defined in Table 1-2)
LDAT  = (testing lengths as defined in Table 1-2 -- fixed lengths)
FDAT  = (all fractions set to 1.0 -- entire program under test)
MDAT  = (all MDAT(I) set to the sum of the CDAT(J), as J goes from
        1 to I-1 -- since the entire program is under test)

NS    = 30
NSB   = 15
NSE   = 29
NSR   = 15
BOPFLG = 1 (Binomial function)
MAXIC = 100
PEC   = 0.85000000000E+00

```

then the call to **SBAMGA** yields:

```

STAT  = 0.5088878882E+02
RFLAG = 0
INDX  = 15
V      = ( 0.2549111310E+01, 0.2846963309E+01, 0.3892940393E+01,
           0.2256273336E+01, 0.4170992533E+01, 0.7605921254E+01,
           0.2370351407E+01, 0.2762384002E+01, 0.1804204025E+01,
           0.2957525316E+01, 0.4264894073E+01, 0.4129487043E+01,
           0.2634130288E+01, 0.4088991420E+01, 0.2554619108E+01)

```

If the **SBAMGD** routine is accessed with:

```

ESF    = 1 (Binomial function)
MAXIC  = 100
NS     = 30
PEC    = 0.85000000000E+00
PED    = 0.3333333333E-01
TNOF   = 0.30600000000E+03
FDAT   = (all fractions set to 1.0 -- entire program under test)
MDAT   = (all MDAT(I) set to the sum of the CDAT(J), as J goes from
        1 to I-1 -- since the entire program is under test)
CDAT   = (fault counts as defined in Table 1-2)
LDAT   = (testing lengths as defined in Table 1-2 -- fixed lengths)

```

then the call to **SBAMGD** yields:

```

STATS  = ( 0.3050010000E+03, 0.8179353590E-01, 0.8500000000E+00)
COUNT = 7
RFLAG  = 0

```

If the SEAMPR routine is accessed with:

NPV = 30 (vector generation)
 NS = 30
 TNOF = 0.3050010000E+03
 PED = 0.8179353590E-01
 PEC = 0.8500000000E+00
 EXP = 0.0000000000E+00
 EXL = 0.0000000000E+00
 ERM = 0.0000000000E+00
 FDAT = (all fractions set to 1.0 -- entire program under test)
 LDAT = (testing lengths as defined in Table 1-2 -- fixed lengths)
 MDAT = (all MDAT(I) set to the sum of the CDAT(J), as J goes from
 1 to I-1 -- since the entire program is under test)

then the call to SEAMPR yields:

PDAT = (0.2494711024E+02,	0.2355662013E+02,	0.2230517903E+02,
	0.2077563991E+02,	0.1924610079E+02,	0.1806418420E+02,
	0.1660416958E+02,	0.1542225299E+02,	0.1410128738E+02,
	0.1291937079E+02,	0.1173745420E+02,	0.1069458661E+02,
	0.9443145513E+01,	0.8469802436E+01,	0.7635508370E+01,
	0.6801214304E+01,	0.6105969249E+01,	0.5549773204E+01,
	0.5202150677E+01,	0.4715479138E+01,	0.4506905622E+01,
	0.4506905622E+01,	0.4298332105E+01,	0.4159283094E+01,
	0.3950709578E+01,	0.3881185072E+01,	0.3881185072E+01,
	0.3881185072E+01,	0.3811660567E+01,	0.3811660567E+01)

If the **SEAMGA** routine is accessed with:

```

CDAT  = (fault counts as defined in Table 1-2)
LDAT  = (testing lengths as defined in Table 1-2 -- fixed lengths)
FDAT  = (all fractions set to 1.0 -- entire program under test)
MDAT  = (all MDAT(I) set to the sum of the CDAT(J), as J goes from
        1 to I-1 -- since the entire program is under test)

NS     = 30
NSB    = 15
NSE    = 29
NSR    = 15
BOPFLG = 2 (Poisson function)
MAXIC  = 100
PEC    = 0.8500000000E+00

```

then the call to **SEAMGA** yields:

```

STAT  = 0.4955744355E+02
RFLAG = 0
INDX  = 15
V     = ( 0.2588726904E+01, 0.2867164853E+01, 0.3873030558E+01,
          0.2278061786E+01, 0.4088018028E+01, 0.7376143698E+01,
          0.2307654620E+01, 0.2635841051E+01, 0.1780318025E+01,
          0.2829297620E+01, 0.4074284657E+01, 0.3947367397E+01,
          0.2536486416E+01, 0.3914100550E+01, 0.2460947391E+01)

```

If the **SEAMGD** routine is accessed with:

```

ESF    = 2 (Poisson function)
MAXIC  = 100
NS     = 30
PEC    = 0.8500000000E+00
PED    = 0.3333333333E-01
TNOF   = 0.3060000000E+03
FDAT   = (all fractions set to 1.0 -- entire program under test)
MDAT   = (all MDAT(I) set to the sum of the CDAT(J), as J goes from
        1 to I-1 -- since the entire program is under test)
CDAT   = (fault counts as defined in Table 1-2)
LDAT   = (testing lengths as defined in Table 1-2 -- fixed lengths)

```

then the call to **SEAMGD** yields:

```

STATS  = ( 0.3050010000E+03, 0.8179170222E-01, 0.8500000000E+00)
COUNT = 7
RFLAG  = 0

```

If the SBAMPR routine is accessed with:

NPV = 30 (vector generation)
 NS = 30
 TNOF = 0.30500100000E+03
 PED = 0.8179170222E-01
 PEC = 0.8500000000E+00
 EXP = 0.0000000000E+00
 EXL = 0.0000000000E+00
 ERM = 0.0000000000E+00
 FDAT = (all fractions set to 1.0 -- entire program under test)
 LDAT = (testing lengths as defined in Table 1-2 -- fixed lengths)
 MDAT = (all MDAT(I) set to the sum of the CDAT(J), as J goes from
 1 to I-1 -- since the entire program is under test)

then the call to SBAMPR yields:

PDAT = (0.2494655097E+02,	0.2355609203E+02,	0.2230467899E+02,
	0.2077517415E+02,	0.1924566932E+02,	0.1806377923E+02,
	0.1660379734E+02,	0.1542190724E+02,	0.1410097125E+02,
	0.1291908116E+02,	0.1173719106E+02,	0.1069434686E+02,
	0.9442933813E+01,	0.8469612556E+01,	0.7635337194E+01,
	0.6801061831E+01,	0.6105832362E+01,	0.5549648787E+01,
	0.5202034053E+01,	0.4715373425E+01,	0.4506804584E+01,
	0.4506804584E+01,	0.4298235743E+01,	0.4159189849E+01,
	0.3950621009E+01,	0.3881098062E+01,	0.3881098062E+01,
	0.3881098062E+01,	0.3811575115E+01,	0.3811575115E+01)

If the SGPOMD routine is accessed with:

```
ESF      = 1 (Maximum Likelihood)
WFSF     = 1 (Weighting function 1)
ALPHA    = 0.0000000000E+00
AN       = 0.0000000000E+00
MAXIC    = 0
CDAT     = (fault counts as defined in Table 1-2)
MDAT     = (all MDAT(I) set to the sum of the CDAT(J), as J goes from
            1 to I-1)
NS       = 30
LDAT     = (testing lengths as defined in Table 1-2 -- fixed lengths)
```

then the call to SGPOMD yields:

```
LDAT     = (testing lengths scaled by indicated Weighting function)
STATS    = ( 0.1877924713E+00, 0.1610658464E+00, 0.2145190962E+00,
            ( 0.3208755511E+03, 0.3167348294E+03, 0.3250162729E+03)
            ( 0.1687555115E+02, 0.1273482941E+02, 0.2101627289E+02)
            ( 0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00)
COUNT   = 1
RFLAG    = 0
```

If the SGPOPR routine is accessed with:

```
NPV      = 3 (expected faults in next interval of testing)
NS       = 30
WFSF     = 1
CDAT     = (fault counts as defined in Table 1-2)
LDAT     = (testing lengths as defined in Table 1-2 -- fixed lengths)
MDAT     = (all MDAT(I) set to the sum of the CDAT(J), as J goes from
            1 to I-1)
EXL      = 0.1000000000E+01
LCOR     = 0.1000000000E+01
PCON     = 0.1877924713E+00
TNOF     = 0.3208755511E+03
ALPHA    = 0.0000000000E+00
```

then the call to SGPOPR yields:

```
PDAT     = ( 0.1490654492E+01, 0.7071284612E+00, 0.2274180522E+01)
```

If the SGPOPR routine is accessed with the following change:

```
NPV      = 30 (vector generation)
```

then the call to SGPOPR yields:

```
PDAT     = ( 0.3012900637E+02, 0.2825108165E+02, 0.2656094941E+02,
            0.2449523223E+02, 0.2242951504E+02, 0.2083327904E+02,
            0.1886145809E+02, 0.1726522208E+02, 0.1548119360E+02,
            0.1388495760E+02, 0.1228872159E+02, 0.1088027806E+02,
            0.9190145816E+01, 0.7875598517E+01, 0.6748843689E+01,
            0.5622088861E+01, 0.4683126504E+01, 0.3931956619E+01,
            0.3462475441E+01, 0.2805201791E+01, 0.2523513084E+01,
            0.2523513084E+01, 0.2241824377E+01, 0.2054031906E+01,
            0.1772343199E+01, 0.1678446963E+01, 0.1678446963E+01,
            0.1678446963E+01, 0.1584550727E+01, 0.1584550727E+01)
```

If the SGPOMA routine is accessed with:

```

CDAT  = (fault counts as defined in Table 1-2)
LDAT  = (testing lengths as defined in Table 1-2 -- fixed lengths)
MDAT  = (all MDAT(I) set to the sum of the CDAT(J), as J goes from
        1 to I-1)

NS    = 30
NSB   = 15
NSE   = 29
NSR   = 15
ALPHA = 0.1000000000E+01

```

then the call to SGPOMA yields:

```

STAT  = 0.4339233957E+02
RFLAG = 0
INDX  = 15
V     = ( 0.2588764205E+01, 0.2867170985E+01, 0.3873031292E+01,
          0.2278086419E+01, 0.4088276529E+01, 0.7376148957E+01,
          0.2307541855E+01, 0.2636672835E+01, 0.1780378666E+01,
          0.2704979647E+01, 0.3399609549E+01, 0.2760331795E+01,
          0.1469074841E+01, 0.2097752164E+01, 0.1164519836E+01)

SDAT  = (testing lengths scaled by indicated Weighting function)

```

If the SGPOMD routine is accessed with:

```

ESF   = 1 (Maximum Likelihood)
WFSF  = 2 (Weighting function 2)
ALPHA = 0.1000000000E+01
AN     = 0.0000000000E+00
MAXIC  = 0
CDAT  = (fault counts as defined in Table 1-2)
MDAT  = (all MDAT(I) set to the sum of the CDAT(J), as J goes from
        1 to I-1)

NS    = 30
LDAT  = (testing lengths as defined in Table 1-2 -- fixed lengths)

```

then the call to SGPOMD yields:

```

LDAT  = (testing lengths scaled by indicated Weighting function)
STATS = ( 0.9389623566E-01, 0.8053292321E-01, 0.1072595481E+00)
        ( 0.3208755511E+03, 0.3179476187E+03, 0.3238034836E+03)
        ( 0.1687555115E+02, 0.1394761873E+02, 0.1980348357E+02)
        ( 0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00)

COUNT = 1
RFLAG  = 0

```

If the SGPOPR routine is accessed with:

```

NPV   = 3 (expected faults in next interval of testing)
NS    = 30
WFSF  = 2
CDAT  = (fault counts as defined in Table 1-2)
LDAT  = (testing lengths as defined in Table 1-2 -- fixed lengths)
MDAT  = (all MDAT(I) set to the sum of the CDAT(J), as J goes from
        1 to I-1)

EXL   = 0.1000000000E+01
LCOR  = 0.1000000000E+01
PCON  = 0.9389623566E-01
TNOP  = 0.3208755511E+03
ALPHA = 0.1000000000E+01

```

then the call to SGPOPR yields:

```

PDAT  = ( 0.1490654492E+01, 0.7071284612E+00, 0.2274180522E+01)

```


If the SGPOPR routine is accessed with the following change:
NPV = 30 (vector generation)

then the call to SGPOPR yields:

PDAT = (0.3012900637E+02,	0.2825108165E+02,	0.2656094941E+02,
	0.2449523223E+02,	0.2242951504E+02,	0.2083327904E+02,
	0.1886145809E+02,	0.1726522208E+02,	0.1548119360E+02,
	0.1388495760E+02,	0.1228872159E+02,	0.1088027806E+02,
	0.9190145816E+01,	0.7875598517E+01,	0.6748843689E+01,
	0.5622088861E+01,	0.4683126504E+01,	0.3931956619E+01,
	0.3462475441E+01,	0.2805201791E+01,	0.2523513084E+01,
	0.2523513084E+01,	0.2241824377E+01,	0.2054031906E+01,
	0.1772343199E+01,	0.1678446963E+01,	0.1678446963E+01,
	0.1678446963E+01,	0.1584550727E+01,	0.1584550727E+01)

If the SGPOMD routine is accessed with:

```

ESP      =      1 (Maximum Likelihood)
WFSF     =      3 (Weighting function 3)
ALPHA    =      0.0000000000E+00
AN       =      0.3060000000E+03
MAXIC    =     100
CDAT     =      (fault counts as defined in Table 1-2)
MDAT     =      (all MDAT(I) set to the sum of the CDAT(J), as J goes from
                  1 to I-1)
NS       =     30
LDAT     =      (testing lengths as defined in Table 1-2 -- variable lengths)

```

then the call to SGPOMD yields:

```

LDAT     =      (testing lengths scaled by indicated Weighting function)
STATS    = (      0.1010656260E+00,      0.0000000000E+00,      0.0000000000E+00)
           (      0.3188095125E+03,      0.0000000000E+00,      0.0000000000E+00)
           (      0.1480951248E+02,      0.0000000000E+00,      0.0000000000E+00)
           (      0.9990161515E-01,      0.0000000000E+00,      0.0000000000E+00)
COUNT   =      8
RFLAG    =      0

```

If the SGPOPR routine is accessed with:

```

NPV      =      1 (expected faults in next interval of testing)
NS       =     30
WFSF     =      3
CDAT     =      (fault counts as defined in Table 1-2)
LDAT     =      (testing lengths as defined in Table 1-2 -- variable lengths)
MDAT     =      (all MDAT(I) set to the sum of the CDAT(J), as J goes from
                  1 to I-1)
EXL      =      0.1000000000E+01
LCOR     =      0.1000000000E+01
PCON     =      0.1010656260E+00
TNOF     =      0.3188095125E+03
ALPHA    =      0.9990161515E-01

```

then the call to SGPOPR yields:

```

PDAT     = (      0.1395667023E+01)

```

If the SGPOPR routine is accessed with the following change:

```

NPV      =     30 (vector generation)

```

then the call to SGPOPR yields:

```

PDAT     = (      0.2805355967E+02,      0.2817893054E+02,      0.2757615455E+02,
                  0.2615674540E+02,      0.2325525106E+02,      0.2072891500E+02,
                  0.1749419106E+02,      0.1714536322E+02,      0.1598827703E+02,
                  0.1473632966E+02,      0.1264939707E+02,      0.1073269161E+02,
                  0.8430732993E+01,      0.7714963793E+01,      0.6855458306E+01,
                  0.5677030084E+01,      0.4695006566E+01,      0.4023373300E+01,
                  0.3518045169E+01,      0.2810585787E+01,      0.2507388909E+01,
                  0.2436352473E+01,      0.2204192031E+01,      0.1945340714E+01,
                  0.1698863901E+01,      0.1597798275E+01,      0.1597798275E+01,
                  0.1597798275E+01,      0.1496732649E+01,      0.1496732649E+01)

```

If the SGPOMD routine is accessed with:

```
ESF      =      2 (Least Squares)
WFSF     =      1 (Weighting function 1)
ALPHA    =      0.0000000000E+00
AN       =      0.0000000000E+00
MAXIC    =      0
CDAT     =      (fault counts as defined in Table 1-2)
MDAT     =      (all MDAT(I) set to the sum of the CDAT(J), as J goes from
                  1 to I-1)
NS       =      30
LDAT     =      (testing lengths as defined in Table 1-2 -- fixed lengths)
```

then the call to SGPOMD yields:

```
LDAT     =      (testing lengths scaled by indicated Weighting function)
STATS    =      ( 0.1454346194E+00, 0.0000000000E+00, 0.0000000000E+00)
              ( 0.3524108196E+03, 0.0000000000E+00, 0.0000000000E+00)
              ( 0.4841081956E+02, 0.0000000000E+00, 0.0000000000E+00)
              ( 0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00)
COUNT   =      1
RFLAG    =      0
```

If the SGPOPR routine is accessed with:

```
NPV      =      1 (expected faults in next interval of testing)
NS       =      30
WFSF     =      1
CDAT     =      (fault counts as defined in Table 1-2)
LDAT     =      (testing lengths as defined in Table 1-2 -- fixed lengths)
MDAT     =      (all MDAT(I) set to the sum of the CDAT(J), as J goes from
                  1 to I-1)
EXL      =      0.1000000000E+01
LCOR     =      0.1000000000E+01
PCON     =      0.1454346194E+00
TNOF     =      0.3524108196E+03
ALPHA    =      0.0000000000E+00
```

then the call to SGPOPR yields:

```
PDAT     =      ( 0.3447587249E+01)
```

If the SGPOPR routine is accessed with the following change:

```
NPV      =      30 (vector generation)
```

then the call to SGPOPR yields:

```
PDAT     =      ( 0.2562636671E+02, 0.2417202052E+02, 0.2286310894E+02,
                  0.2126332813E+02, 0.1966354731E+02, 0.1842735305E+02,
                  0.1690028955E+02, 0.1566409528E+02, 0.1428246640E+02,
                  0.1304627213E+02, 0.1181007787E+02, 0.1071931822E+02,
                  0.9410406646E+01, 0.8392364310E+01, 0.7519756593E+01,
                  0.6647148877E+01, 0.5919975780E+01, 0.5338237302E+01,
                  0.4974650753E+01, 0.4465629585E+01, 0.4247477656E+01,
                  0.4247477656E+01, 0.4029325727E+01, 0.3883891108E+01,
                  0.3665739178E+01, 0.3593021869E+01, 0.3593021869E+01,
                  0.3593021869E+01, 0.3520304559E+01, 0.3520304559E+01)
```

If the SGPOMD routine is accessed with:

```

ESF      =      2 (Least Squares)
WFSF     =      2 (Weighting function 2)
ALPHA    =      0.1000000000E+01
AN       =      0.0000000000E+00
MAXIC    =      0
CDAT     =      (fault counts as defined in Table 1-2)
MDAT     =      (all MDAT(I) set to the sum of the CDAT(J), as J goes from
                  1 to I-1)
NS       =      30
LDAT     =      (testing lengths as defined in Table 1-2 -- fixed lengths)

```

then the call to SGPOMD yields:

```

LDAT     =      (testing lengths scaled by indicated Weighting function)
STATS    =      ( 0.7271730971E-01, 0.0000000000E+00, 0.0000000000E+00)
              ( 0.3524108196E+03, 0.0000000000E+00, 0.0000000000E+00)
              ( 0.4841081956E+02, 0.0000000000E+00, 0.0000000000E+00)
              ( 0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00)
COUNT   =      1
RFLAG    =      0

```

If the SGPOPR routine is accessed with:

```

NPV      =      1 (expected faults in next interval of testing)
NS       =      30
WFSF     =      2
CDAT     =      (fault counts as defined in Table 1-2)
LDAT     =      (testing lengths as defined in Table 1-2 -- fixed lengths)
MDAT     =      (all MDAT(I) set to the sum of the CDAT(J), as J goes from
                  1 to I-1)
EXL      =      0.1000000000E+01
LCOR     =      0.1000000000E+01
PCON     =      0.7271730971E-01
TNOF     =      0.3524108196E+03
ALPHA    =      0.1000000000E+01

```

then the call to SGPOPR yields:

```

PDAT     =      ( 0.3447587249E+01)

```

If the SGPOPR routine is accessed with the following change:

```

NPV      =      30 (vector generation)

```

then the call to SGPOPR yields:

```

PDAT     =      ( 0.2562636671E+02, 0.2417202052E+02, 0.2286310894E+02,
                  0.2126332813E+02, 0.1966354731E+02, 0.1842735305E+02,
                  0.1690028955E+02, 0.1566409528E+02, 0.1428246640E+02,
                  0.1304627213E+02, 0.1181007787E+02, 0.1071931822E+02,
                  0.9410406646E+01, 0.8392364310E+01, 0.7519756593E+01,
                  0.6647148877E+01, 0.5919975780E+01, 0.5338237302E+01,
                  0.4974650753E+01, 0.4465629585E+01, 0.4247477656E+01,
                  0.4247477656E+01, 0.4029325727E+01, 0.3883891108E+01,
                  0.3665739178E+01, 0.3593021869E+01, 0.3593021869E+01,
                  0.3593021869E+01, 0.3520304559E+01, 0.3520304559E+01)

```

If the **SNPIMA** routine is accessed with:

CDAT = (fault counts as defined in Table 1-2)
LDAT = (testing lengths as defined in Table 1-2 -- fixed lengths;
however, modified to reflect cumulative lengths)

NS = 30
NSB = 15
NSE = 29
NSR = 15

then the call to **SNPIMA** yields:

STAT = 0.4306010722E+02
RFLAG = 0
INDX = 15
V = (0.2498235852E+01, 0.2750841260E+01, 0.3711899255E+01,
0.2231553782E+01, 0.3888751256E+01, 0.7287888354E+01,
0.2382679432E+01, 0.2594655742E+01, 0.1779168687E+01,
0.2629793224E+01, 0.3453533898E+01, 0.2908773908E+01,
0.1560401804E+01, 0.2153242990E+01, 0.1228687777E+01)

If the **SNPIMD** routine is accessed with:

ESF = 1 (Maximum Likelihood)
CDAT = (fault counts as defined in Table 1-2)
LDAT = (testing lengths as defined in Table 1-2 -- fixed lengths;
however, modified to reflect cumulative lengths)

NS = 30

then the call to **SNPIMD** yields:

STATS = (0.9700791492E-01, 0.8142694965E-01, 0.1125888802E+00)
(0.3225679527E+03, 0.3050000000E+03, 0.3596171057E+03)
RFLAG = 0

If the **SNPIPR** routine is accessed with:

NPV = 1 (expected faults in the next interval of testing)
NS = 30
TNOF = 0.3225679527E+03
PCON = 0.9700791492E-01
LDAT = (testing lengths as defined in Table 1-2 -- fixed lengths;
however, modified to reflect cumulative lengths)
EXL = 0.1000000000E+01

then the call to **SNPIPR** yields:

PDAT = (0.1624177905E+01)

If the **SNPIPR** routine is accessed with the following change:

NPV = 30 (vector generation)

then the call to **SNPIPR** yields:

PDAT = (0.2982178691E+02, 0.2706472761E+02, 0.2456256169E+02,
0.2229172395E+02, 0.2023082783E+02, 0.1836046398E+02,
0.1666301748E+02, 0.1512250190E+02, 0.1372440880E+02,
0.1245557105E+02, 0.1130403884E+02, 0.1025896715E+02,
0.9310513558E+01, 0.8449745620E+01, 0.7668556692E+01,
0.6959589599E+01, 0.6316167349E+01, 0.5732230243E+01,
0.5202278810E+01, 0.4721322010E+01, 0.4284830233E+01,
0.3888692634E+01, 0.3529178423E+01, 0.3202901724E+01,
0.2906789689E+01, 0.2638053561E+01, 0.2394162404E+01,
0.2172819272E+01, 0.1971939573E+01, 0.1789631438E+01)

If the SNPIND routine is accessed with:

ESF = 2 (Least Squares)
 CDAT = (fault counts as defined in Table 1-2)
 LDAT = (testing lengths as defined in Table 1-2 -- fixed lengths;
 however, modified to reflect cumulative lengths)
 NS = 30

then the call to SNPIND yields:

STATS = (0.7341163266E-01, 0.0000000000E+00, 0.0000000000E+00)
 (0.3572992573E+03, 0.0000000000E+00, 0.0000000000E+00)
 RFLAG = 0

If the SNPIPR routine is accessed with:

NPV = 1 (expected faults in the next interval of testing)
 NS = 30
 TNOF = 0.3572992573E+03
 PCON = 0.7341163266E-01
 LDAT = (testing lengths as defined in Table 1-2 -- fixed lengths;
 however, modified to reflect cumulative lengths)
 EXL = 0.1000000000E+01

then the call to SNPIPR yields:

PDAT = (0.2795666557E+01)

If the SNPIPR routine is accessed with the following change:

NPV = 30 (vector generation)

then the call to SNPIPR yields:

PDAT = (0.2529026502E+02, 0.2350017592E+02, 0.2183679246E+02,
 0.2029114618E+02, 0.1885490344E+02, 0.1752032046E+02,
 0.1628020160E+02, 0.1512786052E+02, 0.1405708415E+02,
 0.1306209920E+02, 0.1213754102E+02, 0.1127842468E+02,
 0.1048011810E+02, 0.9738317051E+01, 0.9049021974E+01,
 0.8408516406E+01, 0.7813346939E+01, 0.7260304606E+01,
 0.6746407575E+01, 0.6268885072E+01, 0.5825162446E+01,
 0.5412847281E+01, 0.5029716504E+01, 0.4673704391E+01,
 0.4342891438E+01, 0.4035494002E+01, 0.3749854693E+01,
 0.3484433434E+01, 0.3237799154E+01, 0.3008622079E+01)

If the SSDWGA routine is accessed with:

CDAT = (fault counts as defined in Table 1-2)
 NS = 30
 NSB = 15
 NSE = 29
 NSR = 15

then the call to SSDWGA yields:

STAT = 0.4306010722E+02
 RFLAG = 0
 INDX = 15
 V = (
 0.2498235852E+01, 0.2750841260E+01, 0.3711899255E+01,
 0.2231553782E+01, 0.3888751256E+01, 0.7287888354E+01,
 0.2382679432E+01, 0.2594655743E+01, 0.1779168687E+01,
 0.2629793224E+01, 0.3453533896E+01, 0.2908773908E+01,
 0.1560401804E+01, 0.2153242989E+01, 0.1228687777E+01)

If the SSDWMD routine is accessed with:

CDAT = (fault counts as defined in Table 1-2)
 NS = 30
 TSF = 1 (Treatment type 1)
 TSS = 1 (use fault counts from all intervals)

then the call to SSDWMD yields:

STATS = (
 0.9700791492E-01, 0.3129164451E+02, 0.3225679527E+03,
 0.4572674899E+02, 0.2119207457E+03, 0.6667156747E+01)
 RFLAG = 0

If the SSDWPR routine is accessed with:

NPV = 1 (expected faults in the next DNOP intervals of testing)
 NS = 30
 DNOP = -0.1000000000E+01
 DNOP = 0.1000000000E+01
 BETA = 0.9700791492E-01
 ALPHA = 0.3129164451E+02
 CDAT = (fault counts as defined in Table 1-2)
 TSF = 1 (Treatment type 1)
 TSS = 1 (use fault counts from all intervals)

then the call to SSDWPR yields:

PDAT = (0.1624177905E+01)
 RFLAG = 0

If the SSDWPR routine is accessed with the following change:

DNOP = 0.2000000000E+01 (expected intervals to find K faults)

then the call to SSDWPR yields:

PDAT = (0.1245897162E+01)
 RFLAG = 0

NSWCDD TR 84-371

If the SSDWPR routine is accessed with the following change:

NPV = 30 (vector generation)

then the call to SSDWPR yields:

PDAT = (0.2982178691E+02,	0.2706472761E+02,	0.2456256169E+02,
	0.2229172395E+02,	0.2023082783E+02,	0.1836046398E+02,
	0.1666301748E+02,	0.1512250190E+02,	0.1372440880E+02,
	0.1245557105E+02,	0.1130403884E+02,	0.1025896715E+02,
	0.9310513558E+01,	0.8449745620E+01,	0.7668556692E+01,
	0.6959589599E+01,	0.6316167349E+01,	0.5732230243E+01,
	0.5202278810E+01,	0.4721322010E+01,	0.4284830233E+01,
	0.3888692634E+01,	0.3529178423E+01,	0.3202901724E+01,
	0.2906789689E+01,	0.2638053561E+01,	0.2394162404E+01,
	0.2172819272E+01,	0.1971939573E+01,	0.1789631438E+01)
RFLAG =	0		

NSWCDD TR 84-371

If the SEDWED routine is accessed with:

CDAT = (fault counts as defined in Table 1-2)
NS = 30
TSF = 2 (Treatment type 2)
TSS = 3 (ignore fault counts from first two intervals)

then the call to SEDWED yields:

STATS = (0.1118482797E+00, 0.3122624387E+02, 0.2791839441E+03,
0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00)
RFLAG = 4

If the SSDWMD routine is accessed with:

CDAT = (fault counts as defined in Table 1-2)
 NS = 30
 TSF = 3 (Treatment type 3)
 TSS = 3 (combine fault counts from first two intervals)

then the call to SSDWMD yields:

STATS = (0.9700285891E-01, 0.3147499057E+02, 0.3244748755E+03,
 0.3770143892E+02, 0.1598259424E+03, 0.8181515897E+01)
 RFLAG = 0

If the SSDWPR routine is accessed with:

NPV = 1 (expected faults in the next DNOP intervals of testing)
 NS = 30
 DNOP = -0.1000000000E+01
 DNOP = 0.1000000000E+01
 BETA = 0.9700285891E-01
 ALPHA = 0.3129028700E+02
 CDAT = (fault counts as defined in Table 1-2)
 TSF = 3 (Treatment type 3)
 TSS = 3 (combine fault counts from first two intervals)

then the call to SSDWPR yields:

PDAT = (0.1633946277E+01)
 RFLAG = 0

If the SSDWPR routine is accessed with the following change:

DNOP = 0.2000000000E+01 (expected intervals to find K faults)

then the call to SSDWPR yields:

PDAT = (0.2117091150E+01)
 RFLAG = 0

If the SSDWPR routine is accessed with the following change:

NPV = 30 (vector generation)

then the call to SSDWPR yields:

PDAT = (0.2999659529E+02, 0.2722351239E+02, 0.2470679154E+02,
 0.2242273295E+02, 0.2034982779E+02, 0.1846855563E+02,
 0.1676120067E+02, 0.1521168485E+02, 0.1380541648E+02,
 0.1252915282E+02, 0.1137087539E+02, 0.1031967675E+02,
 0.9365657840E+01, 0.8499834724E+01, 0.7714054001E+01,
 0.7000916025E+01, 0.6353705222E+01, 0.5766326850E+01,
 0.5233249605E+01, 0.4749453532E+01, 0.4310382756E+01,
 0.3911902575E+01, 0.3550260528E+01, 0.3222051054E+01,
 0.2924183426E+01, 0.2653852644E+01, 0.2408513021E+01,
 0.2185854209E+01, 0.1983779444E+01)
 RFLAG = 0

If the **SESHMA** routine is accessed with:

CDAT = (fault counts as defined in Table 1-2)
 LDAT = (testing lengths as defined in Table 1-2 -- fixed lengths;
 however, modified to reflect cumulative lengths)
 NS = 30
 NSB = 15
 NSE = 29
 NSR = 15

then the call to **SESHMA** yields:

STAT = 0.2503691586E+02
 RFLAG = 0
 INDX = 15
 V = (0.2439316410E+01, 0.2070009411E+01, 0.1821933189E+01,
 0.2271850504E+01, 0.1755653720E+01, 0.3628892295E+01,
 0.1504368198E+01, 0.1333611297E+01, 0.1742837425E+01,
 0.1160112845E+01, 0.1368270641E+01, 0.1083053966E+01,
 0.1010823117E+01, 0.7150368013E+00, 0.1131146036E+01)

If the **SESHMD** routine is accessed with:

CDAT = (fault counts as defined in Table 1-2)
 LDAT = (testing lengths as defined in Table 1-2 -- fixed lengths;
 however, modified to reflect cumulative lengths)
 NS = 30

then the call to **SESHMD** yields:

STATS = (0.2266401535E+00, 0.2071833397E+00, 0.2460969674E+00)
 (0.3076747620E+03, 0.3050000000E+03, 0.3420561298E+03)
 RFLAG = 0

If the **SESHPR** routine is accessed with:

NPV = 1 (expected faults in the next interval of testing)
 NS = 30
 TNOF = 0.3076747620E+03
 PCON = 0.2266401535E+00
 LDAT = (testing lengths as defined in Table 1-2 -- fixed lengths;
 however, modified to reflect cumulative lengths)
 EXL = 0.1000000000E+01

then the call to **SESHPR** yields:

PDAT = (0.4804568745E+00)

If the **SESHPR** routine is accessed with the following change:

NPV = 2 (probability of execution without fault in next interval)

then the call to **SESHPR** yields:

PDAT = (0.6185007501E+00)

If the SESNPR routine is accessed with the following change:

NPV = 30 (vector generation)

then the call to SESNPR yields:

PDAT = (0.6803656142E+01,	0.1669724626E+02,	0.2229834798E+02,
	0.2494105782E+02,	0.2559491061E+02,	0.2495787573E+02,
	0.2352662805E+02,	0.2164948611E+02,	0.1956615755E+02,
	0.1743746553E+02,	0.1536748517E+02,	0.1341994585E+02,
	0.1163031478E+02,	0.1001463885E+02,	0.8575962657E+01,
	0.7308942066E+01,	0.6203121387E+01,	0.5245226299E+01,
	0.4420737037E+01,	0.3714939430E+01,	0.3113600867E+01,
	0.2603380102E+01,	0.2172051045E+01,	0.1808599122E+01,
	0.1503232702E+01,	0.1247340061E+01,	0.1033413505E+01,
	0.8549557518E+00,	0.7063788709E+00,	0.5829026667E+00)

APPENDIX C
EXAMPLES
FOR UTILITY ROUTINE
USAGES

If the SMFEDT routine is accessed with:

```

PRC   = ( 1 , 1 , 21 ,
          0 )
ADA   = ( 0.3100000000E+02, 0.0000000000E+00, 0.0000000000E+00)
        ( 0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00)
ASZ   = 3
DSZ   = 1000
DT1   = ( 0.2000000000E+01, 0.1000000000E+01, 0.2500000000E+01,
          0.3000000000E+01, 0.1500000000E+01, 0.2500000000E+01,
          0.4000000000E+01, 0.6000000000E+01, 0.6500000000E+01,
          0.8000000000E+01, 0.7000000000E+01, 0.9500000000E+01,
          0.1350000000E+02, 0.1100000000E+02, 0.1500000000E+02,
          0.1800000000E+02, 0.1200000000E+02, 0.2250000000E+02,
          0.2500000000E+02, 0.2100000000E+02, 0.3000000000E+02,
          0.2800000000E+02, 0.3650000000E+02, 0.4200000000E+02,
          0.5300000000E+02, 0.4800000000E+02, 0.5100000000E+02,
          0.4750000000E+02, 0.5100000000E+02, 0.3000000000E+02)
DT2   = ( 0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00)
USZ   = 30

```

then the call to SMFEDT yields:

```

DT1   = ( 0.2000000000E+01, 0.1000000000E+01, 0.2500000000E+01,
          0.3000000000E+01, 0.1500000000E+01, 0.2500000000E+01,
          0.4000000000E+01, 0.6000000000E+01, 0.6500000000E+01,
          0.8000000000E+01, 0.7000000000E+01, 0.9500000000E+01,
          0.1350000000E+02, 0.1100000000E+02, 0.1500000000E+02,
          0.1800000000E+02, 0.1200000000E+02, 0.2250000000E+02,
          0.2500000000E+02, 0.2100000000E+02, 0.3100000000E+02,
          0.2800000000E+02, 0.3650000000E+02, 0.4200000000E+02,
          0.5300000000E+02, 0.4800000000E+02, 0.5100000000E+02,
          0.4750000000E+02, 0.5100000000E+02, 0.3000000000E+02)
DT2   = ( 0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
          0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00)
USZ   = 30
ERRFLG = 0

```

If the SMFEDT routine is accessed with:

```

PRC      = (      2      ,      1      ,      10      ,
              15      )
ADA      = (      0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00)
              (      0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00)
ASZ      =      3
DSZ      =      1000
DT1      = (      0.2000000000E+01,      0.1000000000E+01,      0.2500000000E+01,
              0.3000000000E+01,      0.1500000000E+01,      0.2500000000E+01,
              0.4000000000E+01,      0.6000000000E+01,      0.6500000000E+01,
              0.8000000000E+01,      0.7000000000E+01,      0.9500000000E+01,
              0.1350000000E+02,      0.1100000000E+02,      0.1500000000E+02,
              0.1800000000E+02,      0.1200000000E+02,      0.2250000000E+02,
              0.2500000000E+02,      0.2100000000E+02,      0.3100000000E+02,
              0.2800000000E+02,      0.3650000000E+02,      0.4200000000E+02,
              0.5300000000E+02,      0.4800000000E+02,      0.5100000000E+02,
              0.4750000000E+02,      0.5100000000E+02,      0.3000000000E+02)
DT2      = (      0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00)
USZ      =      30

```

then the call to SMFEDT yields:

```

DT1      = (      0.2000000000E+01,      0.1000000000E+01,      0.2500000000E+01,
              0.3000000000E+01,      0.1500000000E+01,      0.2500000000E+01,
              0.4000000000E+01,      0.6000000000E+01,      0.6500000000E+01,
              0.1800000000E+02,      0.1200000000E+02,      0.2250000000E+02,
              0.2500000000E+02,      0.2100000000E+02,      0.3100000000E+02,
              0.2800000000E+02,      0.3650000000E+02,      0.4200000000E+02,
              0.5300000000E+02,      0.4800000000E+02,      0.5100000000E+02,
              0.4750000000E+02,      0.5100000000E+02,      0.3000000000E+02)
DT2      = (      0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00,
              0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00)
USZ      =      24
ERRFLG   =      0

```

If the SMFEDT routine is accessed with:

```

PRC      = (      3      ,      2      ,      19      ,
              3      )
ADA      = (      0.6000000000E+01,      0.5000000000E+01,      0.4000000000E+01)
              (      0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01)
ASZ      =      3
DSZ      =      1000
DT1      = (      0.2000000000E+02,      0.1800000000E+02,      0.2200000000E+02,
              0.2200000000E+02,      0.1700000000E+02,      0.2100000000E+02,
              0.1700000000E+02,      0.1900000000E+02,      0.1700000000E+02,
              0.1700000000E+02,      0.1500000000E+02,      0.1800000000E+02,
              0.1400000000E+02,      0.1200000000E+02,      0.1200000000E+02,
              0.1000000000E+02,      0.8000000000E+01,      0.5000000000E+01,
              0.7000000000E+01,      0.3000000000E+01,      0.0000000000E+00,
              0.3000000000E+01,      0.2000000000E+01,      0.3000000000E+01,
              0.1000000000E+01,      0.0000000000E+00,      0.0000000000E+00,
              0.1000000000E+01,      0.0000000000E+00,      0.1000000000E+01)
DT2      = (      0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01)
USZ      =      30

```

then the call to SMFEDT yields:

```

DT1      = (      0.2000000000E+02,      0.1800000000E+02,      0.2200000000E+02,
              0.2200000000E+02,      0.1700000000E+02,      0.2100000000E+02,
              0.1700000000E+02,      0.1900000000E+02,      0.1700000000E+02,
              0.1700000000E+02,      0.1500000000E+02,      0.1800000000E+02,
              0.1400000000E+02,      0.1200000000E+02,      0.1200000000E+02,
              0.1000000000E+02,      0.8000000000E+01,      0.5000000000E+01,
              0.7000000000E+01,      0.6000000000E+01,      0.5000000000E+01,
              0.4000000000E+01,      0.3000000000E+01,      0.0000000000E+00,
              0.3000000000E+01,      0.2000000000E+01,      0.3000000000E+01,
              0.1000000000E+01,      0.0000000000E+00,      0.0000000000E+00,
              0.1000000000E+01,      0.0000000000E+00,      0.1000000000E+01)
DT2      = (      0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
              0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01)
USZ      =      33
ERRFLG   =      0

```


If the SMFEDT routine is accessed with:

```

PRC      = (      4      ,      2      ,      10      ,
              13      )
ADA      = (      0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00)
              (      0.0000000000E+00,      0.0000000000E+00,      0.0000000000E+00)
ASZ      =      3
DSZ      =      1000
DT1      = (      0.2000000000E+02,      0.1800000000E+02,      0.2200000000E+02,
                  0.2200000000E+02,      0.1700000000E+02,      0.2100000000E+02,
                  0.1700000000E+02,      0.1900000000E+02,      0.1700000000E+02,
                  0.1700000000E+02,      0.1500000000E+02,      0.1800000000E+02,
                  0.1400000000E+02,      0.1200000000E+02,      0.1200000000E+02,
                  0.1000000000E+02,      0.8000000000E+01,      0.5000000000E+01,
                  0.7000000000E+01,      0.6000000000E+01,      0.5000000000E+01,
                  0.4000000000E+01,      0.3000000000E+01,      0.0000000000E+00,
                  0.3000000000E+01,      0.2000000000E+01,      0.3000000000E+01,
                  0.1000000000E+01,      0.0000000000E+00,      0.0000000000E+00,
                  0.1000000000E+01,      0.0000000000E+00,      0.1000000000E+01)
DT2      = (      0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01)
USZ      =      33

```

then the call to SMFEDT yields:

```

DT1      = (      0.2000000000E+02,      0.1800000000E+02,      0.2200000000E+02,
                  0.2200000000E+02,      0.1700000000E+02,      0.2100000000E+02,
                  0.1700000000E+02,      0.1900000000E+02,      0.1700000000E+02,
                  0.6400000000E+02,      0.1200000000E+02,      0.1200000000E+02,
                  0.1000000000E+02,      0.8000000000E+01,      0.5000000000E+01,
                  0.7000000000E+01,      0.6000000000E+01,      0.5000000000E+01,
                  0.4000000000E+01,      0.3000000000E+01,      0.0000000000E+00,
                  0.3000000000E+01,      0.2000000000E+01,      0.3000000000E+01,
                  0.1000000000E+01,      0.0000000000E+00,      0.0000000000E+00,
                  0.1000000000E+01,      0.0000000000E+00,      0.1000000000E+01)
DT2      = (      0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01,
                  0.1000000000E+01,      0.1000000000E+01,      0.1000000000E+01)
USZ      =      30
ERRFLG   =      0

```

If the SMFTRN routine is accessed with:

```

NS      =      30
A       =      0.1000000000E+02
B       =      0.5000000000E+01
TYPE    =      1
DAT     = (    0.2000000000E+01,    0.1000000000E+01,    0.2500000000E+01,
               0.3000000000E+01,    0.1500000000E+01,    0.2500000000E+01,
               0.4000000000E+01,    0.6000000000E+01,    0.6500000000E+01,
               0.8000000000E+01,    0.7000000000E+01,    0.9500000000E+01,
               0.1350000000E+02,    0.1100000000E+02,    0.1500000000E+02,
               0.1800000000E+02,    0.1200000000E+02,    0.2250000000E+02,
               0.2500000000E+02,    0.2100000000E+02,    0.3000000000E+02,
               0.2800000000E+02,    0.3650000000E+02,    0.4200000000E+02,
               0.5300000000E+02,    0.4800000000E+02,    0.5100000000E+02,
               0.4750000000E+02,    0.5100000000E+02,    0.3000000000E+02)

```

then the call to SMFTRN yields:

```

DAT     = (    0.3218875825E+01,    0.2708050201E+01,    0.3401197382E+01,
               0.3555348061E+01,    0.2995732274E+01,    0.3401197382E+01,
               0.3806662490E+01,    0.4174387270E+01,    0.4248495242E+01,
               0.4442651256E+01,    0.4317488114E+01,    0.4605170186E+01,
               0.4941642423E+01,    0.4744932128E+01,    0.5043425117E+01,
               0.5220355825E+01,    0.4828313737E+01,    0.5438079309E+01,
               0.5541263545E+01,    0.5370638028E+01,    0.5720311777E+01,
               0.5652489180E+01,    0.5913503006E+01,    0.6052089169E+01,
               0.6282266747E+01,    0.6184148891E+01,    0.6244166901E+01,
               0.6173786104E+01,    0.6244166901E+01,    0.5720311777E+01)
ERRFLG =      0

```

If the SMFTRN routine is accessed with:

```

NS      =      30
A       =      0.5000000000E+00
B       =      0.1000000000E+01
TYPE    =      2
DAT     = (    0.2000000000E+01,    0.1000000000E+01,    0.2500000000E+01,
               0.3000000000E+01,    0.1500000000E+01,    0.2500000000E+01,
               0.4000000000E+01,    0.6000000000E+01,    0.6500000000E+01,
               0.8000000000E+01,    0.7000000000E+01,    0.9500000000E+01,
               0.1350000000E+02,    0.1100000000E+02,    0.1500000000E+02,
               0.1800000000E+02,    0.1200000000E+02,    0.2250000000E+02,
               0.2500000000E+02,    0.2100000000E+02,    0.3000000000E+02,
               0.2800000000E+02,    0.3650000000E+02,    0.4200000000E+02,
               0.5300000000E+02,    0.4800000000E+02,    0.5100000000E+02,
               0.4750000000E+02,    0.5100000000E+02,    0.3000000000E+02)

```

then the call to SMFTRN yields:

```

DAT     = (    0.7389056099E+01,    0.4481689070E+01,    0.9487735836E+01,
               0.1218249396E+02,    0.5754602676E+01,    0.9487735836E+01,
               0.2008553692E+02,    0.5459815003E+02,    0.7010541235E+02,
               0.1484131591E+03,    0.9001713130E+02,    0.3141906603E+03,
               0.2321572415E+04,    0.6651416330E+03,    0.4914768840E+04,
               0.2202646579E+05,    0.1096633158E+04,    0.2089812889E+06,
               0.7294163698E+06,    0.9871577101E+05,    0.8886110521E+07,
               0.3269017372E+07,    0.2291758109E+09,    0.3584912846E+10,
               0.8771992513E+12,    0.7200489934E+11,    0.3227035704E+12,
               0.5607747199E+11,    0.3227035704E+12,    0.8886110521E+07)
ERRFLG =      0

```

If the SMFTRN routine is accessed with:

```

NS      = 30
A       = 0.1666666667E-01
B       = 0.0000000000E+00
TYPE    = 5
DAT     = ( 0.2000000000E+01, 0.1000000000E+01, 0.2500000000E+01,
            0.3000000000E+01, 0.1500000000E+01, 0.2500000000E+01,
            0.4000000000E+01, 0.6000000000E+01, 0.6500000000E+01,
            0.8000000000E+01, 0.7000000000E+01, 0.9500000000E+01,
            0.1350000000E+02, 0.1100000000E+02, 0.1500000000E+02,
            0.1800000000E+02, 0.1200000000E+02, 0.2250000000E+02,
            0.2500000000E+02, 0.2100000000E+02, 0.3000000000E+02,
            0.2800000000E+02, 0.3650000000E+02, 0.4200000000E+02,
            0.5300000000E+02, 0.4800000000E+02, 0.5100000000E+02,
            0.4750000000E+02, 0.5100000000E+02, 0.3000000000E+02)

```

then the call to SMFTRN yields:

```

DAT     = ( 0.3333333333E-01, 0.1666666667E-01, 0.4166666667E-01,
            0.5000000000E-01, 0.2500000000E-01, 0.4166666667E-01,
            0.6666666667E-01, 0.1000000000E+00, 0.1083333333E+00,
            0.1333333333E+00, 0.1166666667E+00, 0.1583333333E+00,
            0.2250000000E+00, 0.1833333333E+00, 0.2500000000E+00,
            0.3000000000E+00, 0.2000000000E+00, 0.3750000000E+00,
            0.4166666667E+00, 0.3500000000E+00, 0.5000000000E+00,
            0.4666666667E+00, 0.6083333333E+00, 0.7000000000E+00,
            0.8833333333E+00, 0.8000000000E+00, 0.8500000000E+00,
            0.7916666667E+00, 0.8500000000E+00, 0.5000000000E+00)

ERRFLG = 0

```

If the SMFTRN routine is accessed with:

```

NS      = 29
A       = 0.0000000000E+00
B       = 0.0000000000E+00
TYPE    = 6
DAT     = ( 0.2000000000E+01, 0.1000000000E+01, 0.2500000000E+01,
            0.3000000000E+01, 0.1500000000E+01, 0.2500000000E+01,
            0.4000000000E+01, 0.6000000000E+01, 0.6500000000E+01,
            0.8000000000E+01, 0.7000000000E+01, 0.9500000000E+01,
            0.1350000000E+02, 0.1100000000E+02, 0.1500000000E+02,
            0.1800000000E+02, 0.1200000000E+02, 0.2250000000E+02,
            0.2500000000E+02, 0.2100000000E+02, 0.3000000000E+02,
            0.2800000000E+02, 0.3650000000E+02, 0.4200000000E+02,
            0.5300000000E+02, 0.4800000000E+02, 0.5100000000E+02,
            0.4750000000E+02, 0.5100000000E+02, )

```

then the call to SMFTRN yields:

```

DAT     = ( 0.2000000000E+01, 0.3000000000E+01, 0.5500000000E+01,
            0.8500000000E+01, 0.1000000000E+02, 0.1250000000E+02,
            0.1650000000E+02, 0.2250000000E+02, 0.2900000000E+02,
            0.3700000000E+02, 0.4400000000E+02, 0.5350000000E+02,
            0.6700000000E+02, 0.7800000000E+02, 0.9300000000E+02,
            0.1110000000E+03, 0.1230000000E+03, 0.1455000000E+03,
            0.1705000000E+03, 0.1915000000E+03, 0.2215000000E+03,
            0.2495000000E+03, 0.2860000000E+03, 0.3280000000E+03,
            0.3810000000E+03, 0.4290000000E+03, 0.4800000000E+03,
            0.5275000000E+03, 0.5785000000E+03)

ERRFLG = 0

```

If the SMFDST routine is accessed with:

```

NS      = 29
NST     = 11
DAT     = ( 0.2000000000E+01, 0.1000000000E+01, 0.2500000000E+01,
            0.3000000000E+01, 0.1500000000E+01, 0.2500000000E+01,
            0.4000000000E+01, 0.6000000000E+01, 0.6500000000E+01,
            0.8000000000E+01, 0.7000000000E+01, 0.9500000000E+01,
            0.1350000000E+02, 0.1100000000E+02, 0.1500000000E+02,
            0.1800000000E+02, 0.1200000000E+02, 0.2250000000E+02,
            0.2500000000E+02, 0.2100000000E+02, 0.3000000000E+02,
            0.2800000000E+02, 0.3650000000E+02, 0.4200000000E+02,
            0.5300000000E+02, 0.4800000000E+02, 0.5100000000E+02,
            0.4750000000E+02, 0.5100000000E+02)

```

then the call to SMFDST yields:

```

DAT     = ( 0.1000000000E+01, 0.1500000000E+01, 0.2000000000E+01,
            0.2500000000E+01, 0.2500000000E+01, 0.3000000000E+01,
            0.4000000000E+01, 0.6000000000E+01, 0.6500000000E+01,
            0.7000000000E+01, 0.8000000000E+01, 0.9500000000E+01,
            0.1100000000E+02, 0.1200000000E+02, 0.1350000000E+02,
            0.1500000000E+02, 0.1800000000E+02, 0.2100000000E+02,
            0.2250000000E+02, 0.2500000000E+02, 0.2800000000E+02,
            0.3000000000E+02, 0.3650000000E+02, 0.4200000000E+02,
            0.4750000000E+02, 0.4800000000E+02, 0.5100000000E+02,
            0.5100000000E+02, 0.5300000000E+02)
STATS   = ( 0.5785000000E+03, 0.1350000000E+02, 0.6000000000E+01,
            0.3000000000E+02, 0.1000000000E+01, 0.5300000000E+02,
            0.1994827586E+02, 0.1763334140E+02, 0.3109347291E+03,
            0.7008778187E+00, -0.9341055070E+00)

```

If the SMFDST routine is accessed with:

```

NS      = 30
NST     = 11
DAT     = ( 0.2000000000E+02, 0.1800000000E+02, 0.2200000000E+02,
            0.2200000000E+02, 0.1700000000E+02, 0.2100000000E+02,
            0.1700000000E+02, 0.1900000000E+02, 0.1700000000E+02,
            0.1700000000E+02, 0.1500000000E+02, 0.1800000000E+02,
            0.1400000000E+02, 0.1200000000E+02, 0.1200000000E+02,
            0.1000000000E+02, 0.8000000000E+01, 0.5000000000E+01,
            0.7000000000E+01, 0.3000000000E+01, 0.0000000000E+00,
            0.3000000000E+01, 0.2000000000E+01, 0.3000000000E+01,
            0.1000000000E+01, 0.0000000000E+00, 0.0000000000E+00,
            0.1000000000E+01, 0.0000000000E+00, 0.1000000000E+01)

```

then the call to SMFDST yields:

```

DAT     = ( 0.0000000000E+00, 0.0000000000E+00, 0.0000000000E+00,
            0.0000000000E+00, 0.1000000000E+01, 0.1000000000E+01,
            0.1000000000E+01, 0.2000000000E+01, 0.3000000000E+01,
            0.3000000000E+01, 0.3000000000E+01, 0.5000000000E+01,
            0.7000000000E+01, 0.8000000000E+01, 0.1000000000E+02,
            0.1200000000E+02, 0.1200000000E+02, 0.1400000000E+02,
            0.1500000000E+02, 0.1700000000E+02, 0.1700000000E+02,
            0.1700000000E+02, 0.1700000000E+02, 0.1800000000E+02,
            0.1800000000E+02, 0.1900000000E+02, 0.2000000000E+02,
            0.2100000000E+02, 0.2200000000E+02, 0.2200000000E+02)
STATS   = ( 0.3050000000E+03, 0.1100000000E+02, 0.2000000000E+01,
            0.1700000000E+02, 0.0000000000E+00, 0.2200000000E+02,
            0.1016666667E+02, 0.7996047874E+01, 0.6393678161E+02,
            0.2070259230E-01, -0.1581327368E+01)

```

If the SMFGOF routine is accessed with:

DAT = (fault counts as defined in Table 1-2)
 PDAT = (predictions from the Brooks and Motley model)
 - Binomial function -
 NS = 30
 EST = 2
 CCN = -0.1000000000E+01

then the call to SMFGOF yields:

CHI = 0.4583911260E+02
 DOF = 27
 LIF = 0.3811660567E+01
 RFLAG = 0

If the SMFGOF routine is accessed with the following change:

PDAT = (predictions from the Brooks and Motley model)
 - Poisson function -

then the call to SMFGOF yields:

CHI = 0.4584014104E+02
 DOF = 27
 LIF = 0.3811575115E+01
 RFLAG = 0

If the SMFGOF routine is accessed with:

DAT = (fault counts as defined in Table 1-2)
 PDAT = (predictions from the Generalized Poisson model)
 - Maximum Likelihood - Weighting function 1 -
 NS = 30
 EST = 2
 CCN = -0.1000000000E+01

then the call to SMFGOF yields:

CHI = 0.4322327343E+02
 DOF = 27
 LIF = 0.1584550727E+01
 RFLAG = 0

If the SMFGOF routine is accessed with the following change:

PDAT = (predictions from the Generalized Poisson model)
 - Maximum Likelihood - Weighting function 2 -

then the call to SMFGOF yields:

CHI = 0.4322327343E+02
 DOF = 27
 LIF = 0.1584550727E+01
 RFLAG = 0

If the SMFGOF routine is accessed with the following change:

PDAT = (predictions from the Generalized Poisson model)
 - Least Squares - Weighting function 1 -

then the call to SMFGOF yields:

CHI = 0.4418757516E+02
 DOF = 27
 LIF = 0.3520304559E+01
 RFLAG = 0

If the SMFGOF routine is accessed with the following change:
 PDAT = (predictions from the Generalized Poisson model)
 - Least Squares - Weighting function 2 -

then the call to SMFGOF yields:
 CHI = 0.4418757516E+02
 DOF = 27
 LIF = 0.3520304559E+01
 RFLAG = 0

If the SMFGOF routine is accessed with the following changes:
 PDAT = (predictions from the Generalized Poisson model)
 - Maximum Likelihood - Weighting function 3 -
 EST = 3

then the call to SMFGOF yields:
 CHI = 0.4327795541E+02
 DOF = 26
 LIF = 0.1496732649E+01
 RFLAG = 0

If the SMFGOF routine is accessed with:
 DAT = (fault counts as defined in Table 1-2)
 PDAT = (predictions from the Non-homogeneous Poisson model)
 - Maximum Likelihood -
 NS = 30
 EST = 2
 CCN = -0.1000000000E+01

then the call to SMFGOF yields:
 CHI = 0.4174725412E+02
 DOF = 27
 LIF = 0.1789631438E+01
 RFLAG = 0

If the SMFGOF routine is accessed with the following change:
 PDAT = (predictions from the Non-homogeneous Poisson model)
 - Least Squares -

then the call to SMFGOF yields:
 CHI = 0.4234684566E+02
 DOF = 27
 LIF = 0.3008622079E+01
 RFLAG = 0

If the SMFGOF routine is accessed with:
 DAT = (fault counts as defined in Table 1-2)
 PDAT = (predictions from the Schneidewind model)
 - Treatment type 1 -
 NS = 30
 EST = 2
 CCN = -0.1000000000E+01

then the call to SMFGOF yields:
 CHI = 0.4174725412E+02
 DOF = 27
 LIF = 0.1789631438E+01
 RFLAG = 0

If the SMFGOF routine is accessed with:

DAT = (fault counts as defined in Table 1-2)
PDAT = (predictions from Yamada's S-Shaped Reliability Growth
model)
NS = 30
EST = 2
CCN = -0.1000000000E+01

then the call to SMFGOF yields:

CHI = 0.4647602277E+02
DOF = 27
LIF = 0.5829026667E+00
RFLAG = 0

DISTRIBUTION

	<u>Copies</u>		<u>Copies</u>
DOD ACTIVITIES (CONUS)		NON-DOD ACTIVITIES	
DEFENSE TECHNICAL INFO CENTER	12	ATTN GIFT AND EXCHANGE DIVISION	4
CAMERON STATION		LIBRARY OF CONGRESS	
ALEXANDRIA VA 22304-6145		WASHINGTON DC 20540	
CENTER FOR NAVAL ANALYSIS	1	ATTN MS GLORIA DAVIS	1
4401 FORD AVE		NASA	
ALEXANDRIA VA 22302-0268		COMPUTATIONAL SYS RESEARCH BRANCH	
ATTN CODE E29 (TECHNICAL LIBRARY)	1	AMES RESEARCH CENTER	
COMMANDING OFFICER		MOFFETT FIELD CA 94035-1000	
CSSDD NSWC		ATTN CODE NB2 (MS ALICE LEE)	1
6703 W HIGHWAY 98		NASA	
PANAMA CITY FL 32407-7001		JOHNSON SPACE CENTER	
ATTN CODE 70315 (MR CHUCK KOCH)	1	HOUSTON TX 77058	
NAVAL AIR WARFARE CENTER		ATTN CODE EB42 (MR KEN WILLIAMSON)	1
AIRCRAFT DIVISION WARMINSTER		MARSHALL SPACE FLIGHT CENTER	
WARMINSTER PA 18974		SYSTEMS SOFTWARE BRANCH	
ATTN CODE 2092 (MR BOB DUFRESNE)	1	MARSHALL SPACE FLIGHT CEN AL 35812	
NAVAL UNDERWATER SYSTEMS CENTER		ATTN MS BURDETTE JOYNER	1
BLDG 1259		GODDARD SPACE FLIGHT CENTER	
NEWPORT RI 02841-5047		MISSION OPERATIONS DIVISION	
ATTN AMXSY RM (DR PAUL ELLNER)	1	GREENBELT MD 20771	
DIRECTOR USAMSAA		ATTN CODE NC2 (DR RON NEU)	1
ABERDEEN PROVING GROUNDS MD 21005		NOAA	
ATTN MS BETTY ROE	1	RM 646	
DEFENSE INFO SYSTEMS AGENCY		11400 ROCKVILLE PIKE	
CENTER FOR INFO MANAGEMENT XF		ROCKVILLE MD 20852	
1951 KIDWELL DRIVE 5TH FLOOR		ATTN MR OLIVER SMITH	2
VIENNA VA 22182		EG AND G WASC INC	
ATTN PROF NORMAN SCHNEIDEWIND	1	16156 DAHLGREN ROAD	
NAVAL POSTGRADUATE SCHOOL		DAHLGREN VA 22448	
ADMINISTRATIVE SCIENCES DEPARTMENT			
MONTEREY CA 93943			

DISTRIBUTION (Continued)Copies**INTERNAL**

A10	(SCALZO)	1
B05	(CRISP)	1
B10	(LOREY)	1
B10	(FARR)	10
B40	(HOWELL)	1
B40	(WILSON)	1
B40	(JENKINS)	1
B40	(EDWARDS)	1
B40	(NGUYEN)	1
B40	(HOANG)	1
B40	(LEDERER)	1
E231		3
E232		2
E282	(SWANSBURG)	1
K52	(GALLIER)	1
K52	(ASHTON)	1
L12	(GRIM)	1
L104	(BLACKWELDER)	1
N20	(DUDASH)	1
N23	(MCCONNELL)	1
N23	(BROOKS)	1
N72	(FLEMMINGS)	1
N74	(GIDEP)	1

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 1993		3. REPORT TYPE AND DATES COVERED Final	
4. TITLE AND SUBTITLE Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) Library Access Guide				5. FUNDING NUMBERS	
6. AUTHOR(S) Dr. William H. Farr (B10) Oliver D. Smith (EG&G)					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Surface Warfare Center Dahlgren Division (Code B10) Dahlgren, VA 22448-5000				8. PERFORMING ORGANIZATION REPORT NUMBER NSWCDD TR 84-371 Revision 3	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES					
12a. DISTRIBUTION/AVAILABILITY Approved for public release; distribution is unlimited.				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This is the second in a series of Naval Surface Warfare Center Dahlgren Division (NSWCDD) technical reports concerning software reliability. The first report, <u>A Survey of Software Reliability Modeling and Estimation</u> , NSWC TR 82-171, discusses various approaches advocated for reliability estimation; reviews various models proposed for this estimation process; provides model assumptions, estimates of reliability, and the precision of those estimates; and provides the data required for the models' implementation. Eight software reliability models were selected to form the basis of a library. This library also contains data edit, transformation, general statistics, and Goodness-of-Fit functions. The original Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) Library was described in the 1985 version of this report. The enhanced library, which now contains 11 models and model applicability analyses, is explained herein. The <u>SMERFS User's Guide</u> , NSWCDD TR 84-373, Revision 3, explains the execution of the new SMERFS driver.					
14. SUBJECT TERMS estimation mean-time-between-failures reliability software errors software failures software faults software reliability software reliability models				15. NUMBER OF PAGES 147	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR		